SENSORY KEYS: SECURE COMMUNICATION & MUTUAL AUTHENTICATION USING MODIFIED DIFFIE HELLMAN KEY AGREEMENT SCHEME

by

SRIDHAR MATTA

Submitted in partial fulfilment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2016

This work is dedicated to

My beloved parents, Mr.Kusmeshwar & Mrs. Parvati for their belief in me,

My lovely sister, Miss Sweta, who has grown up to be my best companion

and

To all of my friends, who stood by my side all along.

Table of Contents

LIST OF FIGURES……………………………………………………………….....vi

LIST OF TABLES………………………………………………………………xviii

ABSTRACT.. xix

LIST OF ABBREVIATIONS USED xx

CHAPTER 1 INTRODUCTION.. 1

1.1 BRIEF INTRODUCTION OF TERMS AND CONCEPTS 1

1.1.1 Sensors & Smartphones 1

1.1.2 RSA...3

1.1.3 Diffie Hellman Key Exchange Algorithm 3

1.2 BRIEF INTRODUCTION OF THE PROPOSED APPROACH.. 4

1.2 OUTLINE OF THE THESIS.. 5

CHAPTER 2 BACKGROUND.. 7

2.1 SENSORS... 7

2.2 PHYSICALLY UNCLONABLE FUNCTIONS (PUF) 8

2.3 KEY ALGORITHMS FOR CRYPTOGRAPHY 9

2.4 RSA... 11

2.5 DIFFIE HELLMAN.. 12

2.5.1 MITM ATTACK ON DIFFIE HELLMAN.....................................15

2.8 SECURITY ISSUES.. 16

2.7 SECURITY GOALS 17

CHAPTER 3 RELATED WORK.. 19

3.1 LITERATURE SURVEY ON SMARTPHONE DATA.. 19

3.1.1 SMARTPHONE TRAFFIC ANALYSIS 20

3.1.2 MALWARE DETECTION & PREVENTION 21

3.1.3 USER BEHAVIOR & PATTERNS 22

3.1.4 USER BEHAVIOR PATTERNS 23

3.2 LITERATURE SURVEY ON VARIOUS KEY GENERATION METHODS 24

3.3 LITERATURE SURVEY ON SOME OF THE WORKS ON DIFFIE HELLMAN 27

3.3.1 MITM ATTACK POSSIBILITY in DIFFIE HELLMAN 28

3.3.2 OTHER WORKS ON DIFFIE HELLMAN....................................28

3.4 LITERATURE SURVEY ON BIOMETRIC AND OTHER ALGORITHMS 30

3.4.1 BIOMETRICS for AUTHENTICATION & ENCRYPTION 31
3.4.2 OTHER RELATED WORKS 32
3.5 SUMMARY 33
3.6 MOTIVATION AND RESEARCH OBJECTIVES 34
CHAPTER 4 METHODOLOGY 36
4.1 PROPOSED APPROACH 36
4.2 ARCHITECTURE OF THE PROPOSED APPROACH 37
4.3 APPLICATION USED FOR DATA COLLECTION TO TEST THE PROPOSED APPROACH. 51
CHAPTER 5 IMPLEMENTATION 52
5.1 DEVELOPMENT ENVIRONMENT AND LIBRARIES USED 52
5.1.1 JAVA Environment 52
5.1.2 ANDROID Environment 55
5.2 IMPLEMENTATION DETAILS OF THE PHASES IN THE PROPOSED APPROACH 58
5.2.1 XOR OPERATION. 58
5.2.2 HASH OPERATION 59
5.2.3 KEYED HASH OPERATION 59
5.2.4 SOCKET CONNECTION 60
5.2.5 RANDOM NUMBER GENERATION 61
5.2.6 ‘p’, ‘g’, ‘ACKph’, ‘ACKs’ TERMS GENERATION 61
5.2.7 DIFFIE HELLMAN KEY GENERATION 63
5.2.8 RSA KEY GENERATION, ENCRYPTION & DECRYPTION 64
5.2.9 REFERENCE FRAME GENERATION IN ANDROID APPLICATION 65
5.2.10 PERMISSIONS REQUIRED FOR ANDROID APPLICATION 67
CHAPTER 6 EXPERIMENTAL RESULTS AND ANALYSIS 68
6.1 EXPERIMENTAL SETUP 69
6.2 EVALUATION USING WIRESHARK 71
6.3 EVALUATION USING TEST SCENARIOS 74
6.3.1 USING THE ‘SIMULATED DATA’ 75
6.3.2 WHEN THE DEVICE IS MOVED ‘VERTICALLY’ 79
6.3.3 WHEN THE DEVICE IS IN ‘IDLE’ POSITION 82
6.3.4 WHEN THE USER IS ‘WALKING’ WITH THE DEVICE IN HIS TROUSERS’ POCKET 86

6.3.5 WHEN THE USER IS INSIDE AN 'ELEVATOR' WITH THE DEVICE IN HIS TROUSERS' POCKET....89
6.3.6 WHEN THE USER IS MOVING ON THE 'STAIRS' WITH THE DEVICE IN HIS TROUSERS' POCKET....92
6.3.7 WHEN THE USER IS WITHIN A 'VEHICLE' WITH THE DEVICE IN HIS TROUSERS' POCKET....95
6.3.8 INFERENCES FROM THE TEST CASES....98
6.4 EVALUATION USING SCYTHER....100
6.5 PERFORMANCE ANALYSIS....103
6.6 SECURITY ANALYSIS....107
6.7 SUMMARY OF THE EXPERIMENTAL RESULTS....108
CHAPTER 7 CONCLUSION....110
7.1 LIMITATIONS....113
7.2 DISCUSSION AND FUTURE WORK....113
REFERENCES....115
APPENDIX A....121
APPENDIX B....131
APPENDIX C....141
APPENDIX D....151

LIST OF FIGURES

Figure 1 Evolution of mobile phones........ 2

Figure 2 Sensors in a smartphone........ 7

Figure 3 Key Encryption Techniques........ 9

Figure 4 Digital Signature Process........ 11

Figure 5 Diffie Hellman Algorithm......... 14

Figure 6 Known and unknown entities of Diffie Hellman amongst parties........ 14

Figure 7 MITM in Diffie Hellman........ 16

Figure 8 Overview of the Related works discussion........ 19

Figure 9 Survey on Smartphone Sensors........ 20

Figure 10 Works involving Diffie Hellman Algorithm........ 28

Figure 11 Other related Works........ 31

Figure 12........ 40

Figure 13........ 40

Figure 14........ 41

Figure 15 Registration Phase........ 41

Figure 16 Reference Frame Structure........ 41

Figure 17........ 47

Figure 18........ 47

Figure 19........ 48

Figure 20........ 48

Figure 21........ 49

Figure 22........ 49

Figure 23........ 50

Figure 24 Data Transfer Phase........ 50

Figure 25 Android activity life cycle [55]........ 56

Figure 26 Entities stored at each of the parties involved in the communication........ 68

Figure 27 Encrypted Data during the Registration phase......... 72

Figure 28 Second Message during the Registration phase......... 72

Figure 29 Third Message during the Registration phase......... 72

Figure 30 Traffic content during the Data Transfer phase. 73

Figure 31 First packet transferred in the Data Transfer Phase... 73

Figure 32 Second packet transferred in the Data Transfer Phase 73

Figure 33 Third packet transferred in the Data Transfer Phase 74

Figure 34 Fourth packet transferred in the Data Transfer Phase 74

Figure 35 Execution Time vs ReferenceFrames count & DataFile count for Simulated Data 76

Figure 36 Key values obtained for each Data file with 5 Reference Frames for Simulated Data 76

Figure 37 Key value obtained for each Data file with 10 Reference Frames for Simulated Data 77

Figure 38 Key value obtained for each Data file with 15 Reference Frames for Simulated Data 77

Figure 39 Key value obtained for each Data file with 20 Reference Frames for Simulated Data 78

Figure 40 Key value obtained for each Data file with 25 Reference Frames for Simulated Data 78

Figure 41 Execution Time vs ReferenceFrames count & DataFile count for Vertical movement of the Device 79

Figure 42 Key value obtained for each Data file with 5 Reference Frames for Vertical movement of the Device 80

Figure 43 Key value obtained for each Data file with 10 Reference Frames for Vertical movement of the Device 80

Figure 44 Key value obtained for each Data file with 15 Reference Frames for Vertical movement of the Device 81

Figure 45 Key value obtained for each Data file with 20 Reference Frames for Vertical movement of the Device 81

Figure 46 Key value obtained for each Data file with 25 Reference Frames for Vertical movement of the Device 82

Figure 47 Execution Time vs ReferenceFrames count & DataFile count when the Device is in Idle position 83

Figure 48 Key value obtained for each Data file with 5 Reference Frames when the Device is in Idle position 83
Figure 49 Key value obtained for each Data file with 10 Reference Frames when the Device is in Idle position 84
Figure 50 Key value obtained for each Data file with 15 Reference Frames when the Device is in Idle position 84
Figure 51 Key value obtained for each Data file with 20 Reference Frames when the Device is in Idle position 85
Figure 52 Key value obtained for each Data file with 25 Reference Frames when the Device is in Idle position 85
Figure 53 Execution Time vs ReferenceFrames count & DataFile count when the Device is in trouser pocket while walking 86
Figure 54 Key value obtained for each Data file with 5 Reference Frames when the Device is in trouser pocket while walking 87
Figure 55 Key value obtained for each Data file with 10 Reference Frames when the Device is in trouser pocket while walking 87
Figure 56 Key value obtained for each Data file with 15 Reference Frames when the Device is in trouser pocket while walking 88
Figure 57 Key value obtained for each Data file with 20 Reference Frames when the Device is in trouser pocket while walking 88
Figure 58 Key value obtained for each Data file with 25 Reference Frames when the Device is in trouser pocket while walking 89
Figure 59 Key value obtained with 5 Reference Frames when the Device is in trouser pocket while in elevator 90
Figure 60 Key value obtained with 10 Reference Frames when the Device is in trouser pocket while in elevator 90
Figure 61 Key value obtained with 15 Reference Frames when the Device is in trouser pocket while in elevator 91
Figure 62 Key value obtained 20 Reference Frames when the Device is in trouser pocket while in elevator 91
Figure 63 Key value obtained with 25 Reference Frames. 92

Figure 64 Key value obtained with 5 Reference Frames when the Device is in trouser pocket while walking in stairs 93
Figure 65 Key value obtained with 10 Reference Frames when the Device is in trouser pocket while walking in stairs 93
Figure 66 Key value obtained with 15 Reference Frames when the Device is in trouser pocket while walking in stairs 94
Figure 67 Key value obtained 20 Reference Frames when the Device is in trouser pocket while walking in stairs 94
Figure 68 Key value obtained with 25 Reference Frames when the Device is in trouser pocket while walking in stairs 95
Figure 69 Key value obtained with 5 Reference Frames when the Device is in trouser pocket while moving in vehicle 96
Figure 70 Key value obtained with 10 Reference Frames when the Device is in trouser pocket while moving in vehicle 96
Figure 71 Key value obtained with 15 Reference Frames when the Device is in trouser pocket while moving in vehicle 97
Figure 72 Key value obtained 20 Reference Frames when the Device is in trouser pocket while moving in vehicle 97
Figure 73 Key value obtained with 25 Reference Frames when the Device is in trouser pocket while moving in vehicle 98
Figure 74 Scyther configuration settings. 100
Figure 75 Attacks on Diffie Hellman..................... 101
Figure 76 ‘Ta’ compromised 101
Figure 77 ‘Tb’ compromised 102
Figure 78 Scyther output for Registration Phase. 102
Figure 79 Scyther output for Data Transfer Phase..................... 103
Figure 80 CPU up-time (in milliseconds)..................... 104
Figure 81 CPU process time (in millisec)..................... 105
Figure 82 RAM usage (in bytes)..................... 105
Figure 83 Prevention of MITM using the Proposed Approach 109

Figure 84 Key value obtained with 5 Reference Frames for 5 DataFiles for Simulated Data 121
Figure 85 Key value obtained with 5 Reference Frames for 10 DataFiles for Simulated Data 121
Figure 86 Key value obtained with 5 Reference Frames for 15 DataFiles for Simulated Data 121
Figure 87 Key value obtained with 5 Reference Frames for 20 DataFiles for Simulated Data 122
Figure 88 Key value obtained with 5 Reference Frames for 25 DataFiles for Simulated Data 122
Figure 89 Key value obtained with 5 Reference Frames for 50 DataFiles for Simulated Data 122
Figure 90 Key value obtained with 10 Reference Frames for 5 DataFiles for Simulated Data 123
Figure 91 Key value obtained with 10 Reference Frames for 10 DataFiles for Simulated Data 123
Figure 92 Key value obtained with 10 Reference Frames for 15 DataFiles for Simulated Data 123
Figure 93 Key value obtained with 10 Reference Frames for 20 DataFiles for Simulated Data 124
Figure 94 Key value obtained with 10 Reference Frames for 25 DataFiles for Simulated Data 124
Figure 95 Key value obtained with 10 Reference Frames for 50 DataFiles for Simulated Data 124
Figure 96 Key value obtained with 15 Reference Frames for 5 DataFiles for Simulated Data 125
Figure 97 Key value obtained with 15 Reference Frames for 10 DataFiles for Simulated Data 125
Figure 98 Key value obtained with 15 Reference Frames for 15 DataFiles for Simulated Data 125

Figure 99 Key value obtained with 15 Reference Frames for 20 DataFiles for Simulated Data 126
Figure 100 Key value obtained with 15 Reference Frames for 25 DataFiles for Simulated Data 126
Figure 101 Key value obtained with 15 Reference Frames for 50 DataFiles for Simulated Data 126
Figure 102 Key value obtained with 20 Reference Frames for 5 DataFiles for Simulated Data 127
Figure 103 Key value obtained with 20 Reference Frames for 10 DataFiles for Simulated Data 127
Figure 104 Key value obtained with 20 Reference Frames for 15 DataFiles for Simulated Data 127
Figure 105 Key value obtained with 20 Reference Frames for 20 DataFiles for Simulated Data 128
Figure 106 Key value obtained with 20 Reference Frames for 25 DataFiles for Simulated Data 128
Figure 107 Key value obtained with 20 Reference Frames for 50 DataFiles for Simulated Data 128
Figure 108 Key value obtained with 25 Reference Frames for 5 DataFiles for Simulated Data 129
Figure 109 Key value obtained with 25 Reference Frames for 10 DataFiles for Simulated Data 129
Figure 110 Key value obtained with 25 Reference Frames for 15 DataFiles for Simulated Data 129
Figure 111 Key value obtained with 25 Reference Frames for 20 DataFiles for Simulated Data 130
Figure 112 Key value obtained with 25 Reference Frames for 25 DataFiles for Simulated Data 130
Figure 113 Key value obtained with 25 Reference Frames for 50 DataFiles for Simulated Data 130

Figure 114 Key value obtained with 5 Reference Frames for 5 DataFiles for Vertical movement of the Device 131
Figure 115 Key value obtained with 5 Reference Frames for 10 DataFiles for Vertical movement of the Device 131
Figure 116 Key value obtained with 5 Reference Frames for 15 DataFiles for Vertical movement of the Device. 131
Figure 117 Key value obtained with 5 Reference Frames for 20 DataFiles for Vertical movement of the Device 132
Figure 118 Key value obtained with 5 Reference Frames for 25 DataFiles for Vertical movement of the Device 132
Figure 119 Key value obtained with 5 Reference Frames for 50 DataFiles for Vertical movement of the Device 132
Figure 120 Key value obtained with 10 Reference Frames for 5 DataFiles for Vertical movement of the Device 133
Figure 121 Key value obtained with 10 Reference Frames for 10 DataFiles for Vertical movement of the Device 133
Figure 122 Key value obtained with 10 Reference Frames for 15 DataFiles for Vertical movement of the Device 133
Figure 123 Key value obtained with 10 Reference Frames for 20 DataFiles for Vertical movement of the Device 134
Figure 124 Key value obtained with 10 Reference Frames for 25 DataFiles for Vertical movement of the Device 134
Figure 125 Key value obtained with 10 Reference Frames for 50 DataFiles for Vertical movement of the Device 134
Figure 126 Key value obtained with 15 Reference Frames for 5 DataFiles for Vertical movement of the Device 135
Figure 127 Key value obtained with 15 Reference Frames for 10 DataFiles for Vertical movement of the Device 135
Figure 128 Key value obtained with 15 Reference Frames for 15 DataFiles for Vertical movement of the Device 135

Figure 129 Key value obtained with 15 Reference Frames for 20 DataFiles for Vertical movement of the Device 136
Figure 130 Key value obtained with 15 Reference Frames for 25 DataFiles for Vertical movement of the Device 136
Figure 131 Key value obtained with 15 Reference Frames for 50 DataFiles for Vertical movement of the Device 136
Figure 132 Key value obtained with 20 Reference Frames for 5 DataFiles for Vertical movement of the Device 137
Figure 133 Key value obtained with 20 Reference Frames for 10 DataFiles for Vertical movement of the Device 137
Figure 134 Key value obtained with 20 Reference Frames for 15 DataFiles for Vertical movement of the Device 137
Figure 135 Key value obtained with 20 Reference Frames for 20 DataFiles for Vertical movement of the Device 138
Figure 136 Key value obtained with 20 Reference Frames for 25 DataFiles for Vertical movement of the Device 138
Figure 137 Key value obtained with 20 Reference Frames for 50 DataFiles for Vertical movement of the Device 138
Figure 138 Key value obtained with 25 Reference Frames for 5 DataFiles for Vertical movement of the Device 139
Figure 139 Key value obtained with 25 Reference Frames for 10 DataFiles for Vertical movement of the Device 139
Figure 140 Key value obtained with 25 Reference Frames for 15 DataFiles for Vertical movement of the Device 139
Figure 141 Key value obtained with 25 Reference Frames for 20 DataFiles for Vertical movement of the Device 140
Figure 142 Key value obtained with 25 Reference Frames for 25 DataFiles for Vertical movement of the Device 140
Figure 143 Key value obtained with 25 Reference Frames for 50 DataFiles for Vertical movement of the Device 140

Figure 144 Key value obtained with 5 Reference Frames for 5 DataFiles when the Device is in Idle position 141
Figure 145 Key value obtained with 5 Reference Frames for 10 DataFiles when the Device is in Idle position 141
Figure 146 Key value obtained with 5 Reference Frames for 15 DataFiles when the Device is in Idle position 141
Figure 147 Key value obtained with 5 Reference Frames for 20 DataFiles when the Device is in Idle position 142
Figure 148 Key value obtained with 5 Reference Frames for 25 DataFiles when the Device is in Idle position 142
Figure 149 Key value obtained with 5 Reference Frames for 50 DataFiles when the Device is in Idle position 142
Figure 150 Key value obtained with 10 Reference Frames for 5 DataFiles when the Device is in Idle position 143
Figure 151 Key value obtained with 10 Reference Frames for 10 DataFiles when the Device is in Idle position 143
Figure 152 Key value obtained with 10 Reference Frames for 15 DataFiles when the Device is in Idle position 143
Figure 153 Key value obtained with 10 Reference Frames for 20 DataFiles when the Device is in Idle position 144
Figure 154 Key value obtained with 10 Reference Frames for 25 DataFiles when the Device is in Idle position 144
Figure 155 Key value obtained with 10 Reference Frames for 50 DataFiles when the Device is in Idle position 144
Figure 156 Key value obtained with 15 Reference Frames for 5 DataFiles when the Device is in Idle position 145
Figure 157 Key value obtained with 15 Reference Frames for 10 DataFiles when the Device is in Idle position 145
Figure 158 Key value obtained with 15 Reference Frames for 15 DataFiles when the Device is in Idle position 145

Figure 159 Key value obtained with 15 Reference Frames for 20 DataFiles when the Device is in Idle position 146
Figure 160 Key value obtained with 15 Reference Frames for 25 DataFiles when the Device is in Idle position 146
Figure 161 Key value obtained with 15 Reference Frames for 50 DataFiles when the Device is in Idle position 146
Figure 162 Key value obtained with 20 Reference Frames for 5 DataFiles when the Device is in Idle position 147
Figure 163 Key value obtained with 20 Reference Frames for 10 DataFiles when the Device is in Idle position 147
Figure 164 Key value obtained with 20 Reference Frames for 15 DataFiles when the Device is in Idle position 147
Figure 165 Key value obtained with 20 Reference Frames for 20 DataFiles when the Device is in Idle position 148
Figure 166 Key value obtained with 20 Reference Frames for 25 DataFiles when the Device is in Idle position 148
Figure 167 Key value obtained with 20 Reference Frames for 50 DataFiles when the Device is in Idle position 148
Figure 168 Key value obtained with 25 Reference Frames for 5 DataFiles when the Device is in Idle position 149
Figure 169 Key value obtained with 25 Reference Frames for 10 DataFiles when the Device is in Idle position 149
Figure 170 Key value obtained with 25 Reference Frames for 15 DataFiles when the Device is in Idle position 149
Figure 171 Key value obtained with 25 Reference Frames for 20 DataFiles when the Device is in Idle position 150
Figure 172 Key value obtained with 25 Reference Frames for 25 DataFiles when the Device is in Idle position 150
Figure 173 Key value obtained with 25 Reference Frames for 50 DataFiles when the Device is in Idle position 150

Figure 174 Key value obtained with 5 Reference Frames for 5 DataFiles when the Device is in trouser pocket while walking 151
Figure 175 Key value obtained with 5 Reference Frames for 10 DataFiles when the Device is in trouser pocket while walking 151
Figure 176 Key value obtained with 5 Reference Frames for 15 DataFiles when the Device is in trouser pocket while walking 151
Figure 177 Key value obtained with 5 Reference Frames for 20 DataFiles when the Device is in trouser pocket while walking 152
Figure 178 Key value obtained with 5 Reference Frames for 25 DataFiles when the Device is in trouser pocket while walking 152
Figure 179 Key value obtained with 5 Reference Frames for 50 DataFiles when the Device is in trouser pocket while walking 152
Figure 180 Key value obtained with 10 Reference Frames for 5 DataFiles when the Device is in trouser pocket while walking 153
Figure 181 Key value obtained with 10 Reference Frames for 10 DataFiles when the Device is in trouser pocket while walking 153
Figure 182 Key value obtained with 10 Reference Frames for 15 DataFiles when the Device is in trouser pocket while walking 153
Figure 183 Key value obtained with 10 Reference Frames for 20 DataFiles when the Device is in trouser pocket while walking 154
Figure 184 Key value obtained with 10 Reference Frames for 25 DataFiles when the Device is in trouser pocket while walking 154
Figure 185 Key value obtained with 10 Reference Frames for 50 DataFiles when the Device is in trouser pocket while walking 154
Figure 186 Key value obtained with 15 Reference Frames for 5 DataFiles when the Device is in trouser pocket while walking 155
Figure 187 Key value obtained with 15 Reference Frames for 10 DataFiles when the Device is in trouser pocket while walking 155
Figure 188 Key value obtained with 15 Reference Frames for 15 DataFiles when the Device is in trouser pocket while walking 155

Figure 189 Key value obtained with 15 Reference Frames for 20 DataFiles when the Device is in trouser pocket while walking 156
Figure 190 Key value obtained with 15 Reference Frames for 25 DataFiles when the Device is in trouser pocket while walking 156
Figure 191 Key value obtained with 15 Reference Frames for 50 DataFiles when the Device is in trouser pocket while walking 156
Figure 192 Key value obtained with 20 Reference Frames for 5 DataFiles when the Device is in trouser pocket while walking 157
Figure 193 Key value obtained with 20 Reference Frames for 10 DataFiles when the Device is in trouser pocket while walking 157
Figure 194 Key value obtained with 20 Reference Frames for 15 DataFiles when the Device is in trouser pocket while walking 157
Figure 195 Key value obtained with 20 Reference Frames for 20 DataFiles when the Device is in trouser pocket while walking 158
Figure 196 Key value obtained with 20 Reference Frames for 25 DataFiles when the Device is in trouser pocket while walking 158
Figure 197 Key value obtained with 20 Reference Frames for 50 DataFiles when the Device is in trouser pocket while walking 158
Figure 198 Key value obtained with 25 Reference Frames for 5 DataFiles when the Device is in trouser pocket while walking 159
Figure 199 Key value obtained with 25 Reference Frames for 10 DataFiles when the Device is in trouser pocket while walking 159
Figure 200 Key value obtained with 25 Reference Frames for 15 DataFiles when the Device is in trouser pocket while walking 159
Figure 201 Key value obtained with 25 Reference Frames for 20 DataFiles when the Device is in trouser pocket while walking 160
Figure 202 Key value obtained with 25 Reference Frames for 25 DataFiles when the Device is in trouser pocket while walking 160
Figure 203 Key value obtained with 25 Reference Frames for 50 DataFiles when the Device is in trouser pocket while walking 160

LIST OF TABLES

Table 1 Time taken to brute force the key 12
Table 2 Development environment for the proposed approach 52
Table 3 Platform Specifications 53
Table 4 Application development environment specifications 55

ABSTRACT

Smartphones and Tabloids are becoming the digital entity of identification for every individual. Their portability and programmability have made them a juncture of endless applications. Apart from the numerous gaming apps that are available, applications especially in the fields of health and fitness, and finance often require the data to be transferred to a remote server. Manipulation of that data by a hacker, such as by man in the middle (MITM) attacks can lead to many undesired outcomes. Therefore, secure data transfer is critical in many applications. This research work presents a new variant of the Diffie-Hellman key agreement scheme that uses dynamically changing sensor data to facilitate continuous key updates. Our scheme ensures mutual authentication and mitigates MITM attacks with minimal need for public key infrastructure (PKI). We also propose an access control mechanism that protects data recorded by our application on the phone in case of physical attacks. We have tested the randomness of the keys generated using various real time use-cases. There were no noticeable patterns of key generation or key sequences. We have also evaluated our scheme using a security protocol analyzer tool, 'Scyther'. Our test results have shown that the proposed key agreement scheme is efficient in mitigating MITM attacks.

LIST OF ABBREVIATIONS USED

MITM	Man In The Middle
PKI	Public Key Infrastructure
DH	Diffie Hellman
ECDH	Elliptic Curve Diffie Hellman
NBS	National Bureau of Standards
PUF	Physically Unclonable Function
JVM	Java Virtual Machine
STS	Station to Station Protocol
RSA	Rivest Shamir Adleman
DoS	Denial of Service

CHAPTER 1 INTRODUCTION

We begin the journey through our thesis with this chapter wherein we provide a brief introduction to the various terms used in this work that follows later on. We then present the overview of the proposed approach followed by the gist of the journey that lies ahead.

1.1 BRIEF INTRODUCTION OF TERMS AND CONCEPTS

1.1.1 Sensors & Smartphones

Sensors have become an integral part of everyone’s life in today’s world. Apart from the body sensors which are widely used in health monitoring needs, there are sensors imbibed into various other devices as well [66]. For instance, they are present in vehicles for collision detection, in smartphones for activities such as location detection, orientation detection etc., in wrist watches for tracking the user activities such as running, jogging, pulse etc. The presence of sensors into the devices, which have become an addictive necessity, opened up a plethora of scenarios where it can be used. The health care applications, for example, make use of the sensors manifested in the smartphone and smart watches to constantly monitor the patients’ health conditions remotely and act in advance in-case of unanticipated panic attacks. This not only helps in responding at the dire time of need but also increases the chance of saving the person’s life.

At the same time, Smartphones are becoming more and more pervasive in everyone’s life. The number of smartphone devices sold has drastically increased over the years depicting its expanse into every person’s life [1]. This led to the development of applications which span its wings into every known corner of our daily routine life. There are tons of applications running on the smartphones and tabloids with applications ranging from a simple calculator to applications for playing sophisticated games. Programming languages such as ‘Salesforce’ allows its user to write and debug the code on their smartphones which are later compiled & executed on the cloud server [2].

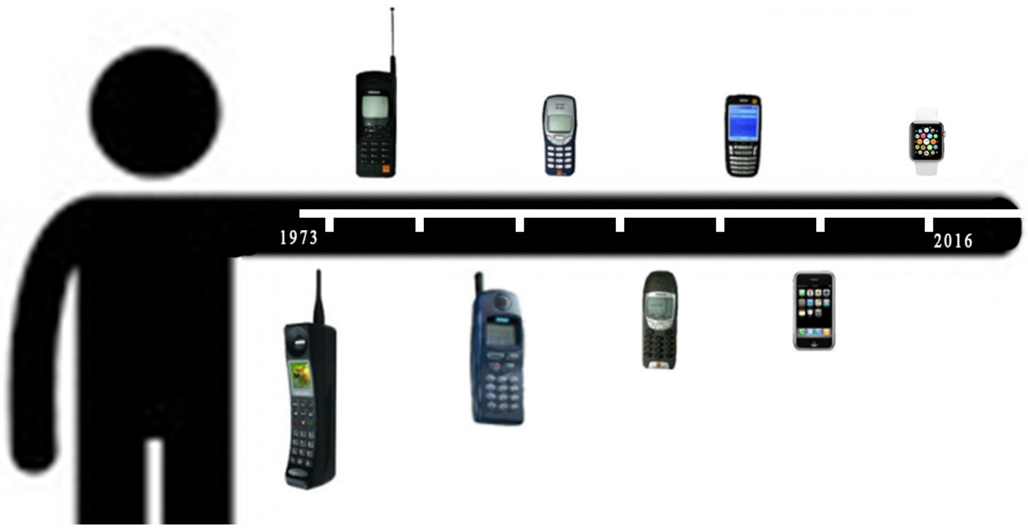


Figure 1 Evolution of mobile phones [Adapted from 60]

Most of the applications require to transfer sensitive data to and fro the device while communicating with the remote server. For instance, Mobile Banking applications authenticate the user using a set of username and password. These credentials are verified in the server and then the concerned access is provided to the user. Thus there is an exchange of the username and the password, though not in raw format, over the communication channel. This data if intercepted and decoded, can lead to disastrous results such as undesired transactions, change of password etc. Few applications which require the use of file transfers are Dropbox and Google Drive which store files in remote servers and/or remote computers and use the AES encryption in the tunnel [3]. AES being a symmetric key requires for a method to transfer the key to the other party in order for successful encryption and decryption to take place. This calls for extra overhead and consumption of extra ounce of energy during the communication [4]. Thus these algorithms which offer top of the state security are high in computation and complexity. The devices such as Smartphones, smart watches etc., being resource constrained, find it challenging to employ such secure algorithms.

1.1.2 RSA

RSA was a replacement for the then lesser secure National Bureau of Standards (NBS) algorithm. It had a fine layout for the PKI. In RSA, the keys used for encryption were public but the keys needed for the decryption process were private i.e. not publicly available. Hence this boils down to the point that only the user who is in possession of the decryption could decrypt the contents of the message, ensuring the confidentiality aspect. These keys were devised in such a way that they were not easily deduced from the publicly available encryption keys. The key pair also helped in generating the Digital Signature which in turn provided authentication. Thus RSA found its role in electronic transactions, especially the electronic fund transaction where the authenticity and the integrity of the message being transferred was required. [7]

1.1.3 Diffie Hellman Key Exchange Algorithm

Diffie Hellman Key Exchange algorithm provided a great solution to the key distribution problem in the PKI. It had a technique where the parties can securely exchange the keys which can later be used for encryption of the contents. This secure exchange was ensured by limiting the exchange of the secret values used for the generation of the keys [5]. Since then many variants of the DH algorithm have come into the picture such as ECDH. But Menezes et.al [6] showed the possibility of the occurrence of the MITM attack in the DH algorithm which proved to be a major setback to the algorithm.

In this thesis, a modified Diffie Hellman approach is proposed which aims at automatic key generation to minimize the probability of occurrence of the MITM attack in the DH algorithm in the smartphone world during its communication with the server or another device. Added to that, it also provides the benefit of avoiding replay attacks with the help of sequence numbers, provision of Access Control with the help of fingerprint and also provides forward & backward secrecy by deleting the key upon reception of an acknowledgement from the recipient.

1.2 Brief introduction of the proposed approach

This research work mainly aims at reducing the probability of MITM and replay attacks with the benefit of access control and forward & backward secrecy using a modified Diffie Hellman approach in a client-server architecture where the client is a smartphone device. The smartphone device communicates with the server via an application. This application performs the required functionalities such as fingerprint detection, sensor data collection etc. and contributes towards accomplishing the above mentioned approach.

The proposed approach functions in two phases. The registration phase in which the values are passed from the smartphone to the server, the data transfer phase in which the data is encrypted and then sent to the server. In the entire lifetime of the mobile application, registration phase happens only once i.e. during the time of installation of the application in the mobile device. It is here, when the required values are collected and sent to the server using the RSA algorithm of PKI. From this point on, in every data transfer phase, all the keys which are used to encrypt the data, are generated using the values which are being exchanged as a part of the message. Hence in short, the registration phase is one time and the data transfer phase is repetitive in the proposed approach.

A combination of the biometric and the PUF of the device is used to provide access control. The biometric authentication, for instance, the fingerprint of the user, is used to provide access control to that user over the application i.e. if the fingerprint fails to find a match, it would not allow the user to access the features of the application, thus preventing him/her from using the application. Similarly, the PUF's of the device are unique in nature. They act as the fingerprint of the device [8]. Thus an amalgam of these two features, which are unique in their own way, are used in this approach to provide access control mechanism. In addition, at the end of each data transfer, two-way acknowledgement is used to ensure the complete reception of the message at the recipient end. Upon reception of the acknowledgments, the keys which were used for encrypting the data are deleted to ensure the forward and backward secrecy along with the assurance that no key can be extracted from the device in case of theft. During the data transfer, sequence numbers are used to prevent the replay attacks from occurring. Sequence numbers are sent as a part of the message which ensures that no data packet is replayed. If the server finds a data packet with a sequence number which is not a successor to the one it already received, it simply drops that particular packet.

The proposed approach generates the keys it uses for the encryption of the data by selecting the sensor values in a random manner. It forms a set of frames each comprising of the sensor values as its fields. Then a random field from a random frame is selected and is then used for the key generation. Since sensors have the capability to capture even the slightest fluctuations, predicting the precise value is relatively hard. This makes it suitable for increasing the unpredictability of the keys being generated thus offering the security we desire against brute-force and other similar attacks which in turn reduces the probability of occurrence of MITM in Diffie Hellman.

The proposed key generation scheme can be applied to any authentication scheme. It need not necessarily be restricted to Diffie Hellman. This research work considers Diffie Hellman as an instance to validate the proposed approach towards key generation. As long as there are sensors or any dynamically fluctuating values such as RSSI values from Access Points, power readings from smart meters in a SmartGrid etc. are available, the proposed technique can be applied. The keys thus generated can be used to encrypt the data in any other algorithm such as RSA, Advanced Encryption Standard (AES). Since the main focus of this research work is to prevent MITM in the Diffie Hellman approach, hence we have limited the implementation and the testing scenarios of this thesis to Diffie Hellman only.

1.2 Outline of the thesis

The organization of the rest of the thesis is as follows – an overview of the PKI, Diffie Hellman, Symmetric & Asymmetric Keys, Smartphone world, Biometric authentication, PUF factors, Sensors in the Smartphones and their functioning is provided in Chapter 2. We then provide a detailed discussion in Chapter 3 on the various ways the sensor data have been used, different techniques implemented for the purpose of encryption along with the existing approaches for resolving and /or mitigating the attacks of Diffie Hellman. Chapter 4 discusses the proposed methodology in detail which is followed by a thorough explanation of the implementation aspect in Chapter 5. Chapter 5 also discusses about the technical specification which influenced the research work. We present the in-depth description of the experiments we conducted and present the evaluated results supporting our proposed approach in Chapter 6. Chapter 7 marks the conclusion of the thesis where we present the limitations and possible enhancements to the proposed approach.

CHAPTER 2 BACKGROUND

In this chapter, a brief overview of the terms and concepts used for the proposed approach are enlisted and explained. Then a short discussion of the existing problems is presented. In short, this chapter, aims at throwing light on the background of the thesis.

2.1 SENSORS

Sensors are the hardware components which detect even the slightest of the variations in the environment surrounding it. The detected change is the output of this sensor. A sensor detects the change in the form of electrical signals for the environment it is designed for, and presents the difference it observes as output, in different forms. For instance, in an accelerometer sensor, out of the numerous parts it consists, capacitance sensor and piezoelectric effect are the important ones. Microscopic crystal structures are used in the piezoelectric effect, which are stressed when the sensor hardware accelerates. This stress is converted into Voltage which is interpreted to determine orientation and/or velocity. The capacitance sensor observes for changes between the capacitances amongst the microstructures which is positioned next to this sensor. The detected change in the capacitance due to an accelerative force experienced by the microstructures, is translated to voltage for interpretation. [9]

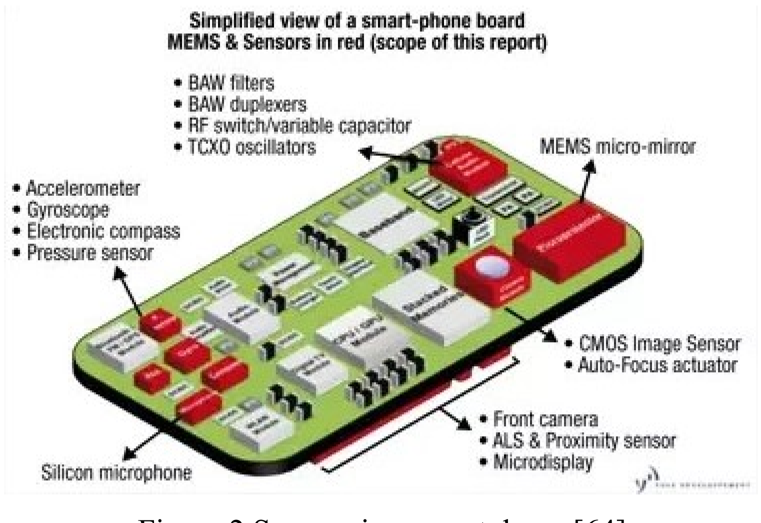


Figure 2 Sensors in a smartphone [64]

Sensors are being used in almost every device . For instance, the mobile devices are having sensors which make the user experience with the device better. The light sensor adjusts the screen brightness automatically, which reduces the need for the user to increase/decrease the screen brightness as and when he enters into a relatively dark/bright environment. Similarly, the accelerometer sensor available in the laptops turns off the hard drive as soon as it detects a sudden free fall. This prevents the reading heads from hitting the hard drive platter and damaging it. Fitbits are designed with sensors which help in monitoring the health of the user by detecting the fluctuations in its sensors and translating them into the number of steps the user has taken, amount of time spent in running etc. [10]

2.2 PHYSICALLY UNCLONABLE FUNCTIONS (PUF)

The PUF are the minute variations that appear in the device hardware due to the change in the temperature of the manufacturing environment, power transmission etc. In short, it acts as the fingerprint for the device. No two devices have the same PUF values. The PUF of a device is relatively easy to derive by the device at a particular instant of time but hard to predict from outside.

At the time of manufacturing of the device, its hardware components such as its Integrated Circuits (IC) are subjected to various electrical and/or physical variations such as electrical interferences, change in temperature, voltage supply etc. These affect the microstructure of the device which makes it virtually Unclonable and unpredictable. Hence for evaluation, a stimulus is applied to the microstructure whose unpredictable reaction is taken and stored as a value. These stimulus reactions change every time it is applied as for each application of the stimulus, the influencing factors such as the voltage, interferences etc. change as well.

Several cryptographic and authentication based algorithms have used PUF for obtaining the desired security. They implement a challenge-response pair which help them authenticate each other. [8] Some have used them for key generation where the delay characteristics are used for generating the keys needed for encryption. [11]

2.3 KEY ALGORITHMS FOR CRYPTOGRAPHY

All the data which is sent over the communication channel has to be encrypted in order to prevent a hacker from interpreting the intercepted message. Encryption is a technique where the plain message is converted into an un-identifiable format. This encrypted data, can only be decrypted with the proper set of key/s. There are two types of Key algorithms. Symmetric Key algorithms and Asymmetric Key algorithms.

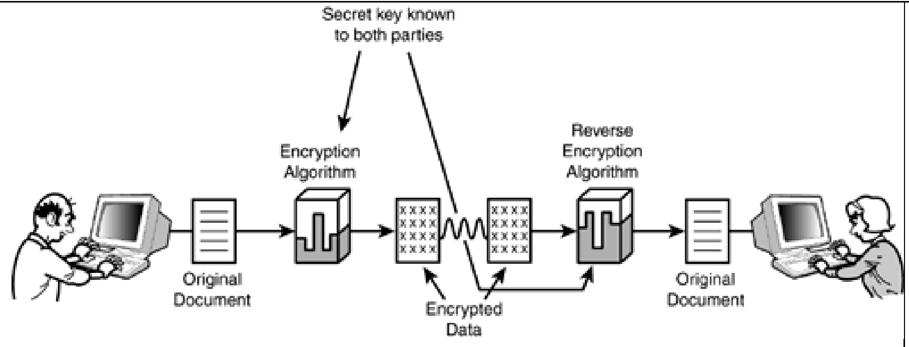
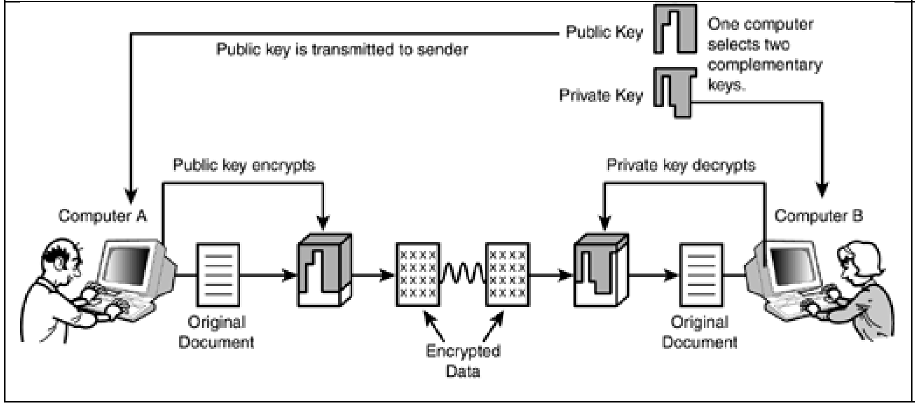
 Secret key known to both parties Encryption Algorithm Reverse Encryption Algorithm Original Document Encrypted Data Original Document	Figure 3 a) Symmetric Key Encryption Technique
 Public key is transmitted to sender Public Key Private Key One computer selects two complementary keys. Public key encrypts Private key decrypts Computer A Computer B Original Document Encrypted Data Original Document	Figure 3 b) Asymmetric Key Encryption Technique

Figure 3 Key Encryption Techniques [61]

Symmetric Key algorithms use the same key/s for encrypting and decrypting the messages. These keys, in practice, represent a shared secret between the two communicating entities. The two parties, exchange the secret key before commencing the data transfer. The sender encrypts the data to be sent using the shared key and sends it into the communication channel. The receiver upon reception, decrypts it using the shared key. This shared secret key is usually used when the parties want to maintain a private session. The only drawback here is the need for both the parties to securely exchange the shared secret and the need to access to the shared secret. Figure 3a illustrates the process behind symmetric key encryption. Symmetric ciphers can be used as Block ciphers or as Stream ciphers. Block ciphers encrypts blocks of bits while Stream ciphers encrypts bit by bit. [12]

Asymmetric Key algorithms, also termed as Public Key Cryptography, uses pairs of keys. Public keys, which is known to every entity involved in the network and Private keys, which is known only to the entity who owns it. Here, the sender encrypts the data using the public key of the receiver and sends it into the communication channel. The receiver, upon reception of the encrypted message, decrypts it using its own private key. It is the level of difficulty for a private key to be deduced or generated from its public key, that decides the strength of the public key cryptography. Thus advertising the public key does not risk the security of the algorithm but leak of the private key of an entity, does compromise the session. Hence, asymmetric key algorithms, does not have the need to exchange shared secrets over a secure channel, like the Symmetric algorithms. But these algorithms are computationally complex and hence consume relatively more resource than the counterpart symmetric algorithms. It is for this reason that the encryption is performed on small blocks of data instead of encrypting the entire chunk of the data which is to be transmitted. Figure 3b illustrates the process behind asymmetric key encryption.

Asymmetric key algorithms are also used in Digital Signature generation. Digital signatures help authenticating the messages. The actual message is hashed initially to generate a ‘digest’. This digest is then encrypted with the private key of the sender to produce the Digital Signature. The receiver upon reception of the digital signature along with the encrypted message, decrypts the digital signature using the public key of the sender. It then hashes the message it obtained after decrypting the encrypted message with its private key. If the former match with the latter, the sender is authenticated. Since hash is a one-way function, although it is possible to decrypt with the sender’s public key, the interceptor cannot deduce the actual message from it nor can he alter the signature with any other digest. SHA-1, SHA-2, MD5 etc. are some of the hashing algorithms which are widely put into use. The process is briefly illustrated in Figure 4.

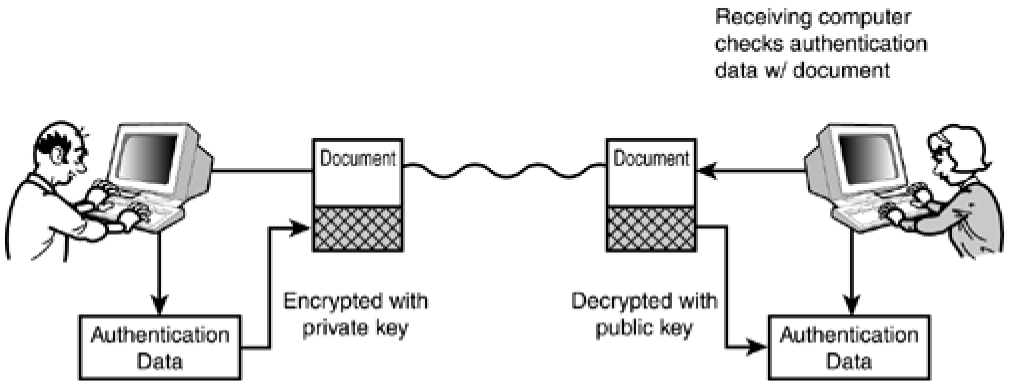

Receiving computer checks authentication data w/ document
Document
Document
Encrypted with private key
Decrypted with public key
Authentication Data
Authentication Data

Figure 4 Digital Signature Process [61]

2.4 RSA

Ron Rivest, Adi Shamir and Leonard Adleman proposed an algorithm in 1978 which implemented Public Key cryptosystem along with Digital Signatures, as a substitute to the then less secure NBS algorithm. RSA was motivated by Diffie Hellman. RSA possess public key cryptosystems i.e. this algorithm does not need a separate courier to transfer the secret key and digital signature i.e. receiver ensures that the message originated from the sender. Each parties have their own private and public keys using which the formerly mentioned public key cryptosystems and digital signature ideas are accomplished.

The sender 'Alice' encrypts the message to be sent to the Receiver 'Bob', using Bob's public key. Bob upon reception of the encrypted message, decrypts it using his own private key. The steps involved in RSA algorithm is as follows;

- Consider two large prime numbers 'p', 'q' and calculate their product as 'n' i.e.

$$n = p * q$$

- Choose an integer 'e' such that it is less than 'n' and is relatively prime to (p – 1)(q – 1) i.e. 'e' and (p – 1)(q – 1) must not have a common factor.
- Find an integer 'd' such that

$$(e * d)\ mod\ \left((p-1)(q-1)\right) = 1$$

- Thus, PUBLIC KEY (e, n) & PRIVATE KEY (d, n) is generated.
- The Encryption formula for message 'M' is

$$C = M^e\ mod\ n$$

- The Decryption formula for encrypted message ‘C’ is

$$M = C^{e} \ mod \ n$$

To show how secure the algorithm is, the authors tried to obtain the keys as a cryptanalyst would attempt. They haven’t considered the scenario where an intruder might attempt to steal the key. The assumption they made was that the time taken by Schroeppel factoring algorithm to compute is one microsecond. Based on the assumption they presented the following [7]:

DIGITS	Number Of Operations	TIME
50	1.4×10^{10}	3.9 hours
75	9.0×10^{12}	104 days
100	2.3×10^{15}	74 years
200	1.2×10^{23}	3.8×10^{9} years
300	1.5×10^{29}	4.9×10^{15} years
500	1.3×10^{39}	4.2×10^{25} years

Table 1 Time taken to brute force the key [7]

Thus the authors recommended a key size of ‘n’ as 200 digits long. The key length can be varied based on the desired speed of encryption versus desired security. [7]

2.5 DIFFIE HELLMAN

Diffie Hellman was first published in 1976 by Whitfield Diffie and Martin E. Hellman but was conceptualized by Ralph Merkle [13]. It is used to provide forward secrecy in Transport Layer Security. Forward security means that; with the compromise of the long term keys does not compromise the keys used in the previous sessions i.e. the encrypted messages sent in the past cannot be decrypted. Thus Diffie Hellman algorithm is a method for secure exchange of the keys over public channel.

Diffie Hellman helps two parties, who have no prior knowledge of each other, to establish the secret key using which the messages are encrypted and decrypted. But here a secure physical channel is required for the first exchange of keys. [14] Diffie Hellman is based on discrete logarithm problem where the original implementation of the algorithm used the cyclic multiplicative groups of integers *mod* p, where p is a prime number and g is a relatively root modulo p. According to Kirchhoff principle, g and p are known by the attackers and are believed not to be secret. Thus the idea was to make two parties communicate each other without having the parties know each other in advance. On the whole, cyclic multiplicative group carries the following property [13]

$$(g^a)^b \, mod \, (p) = (g^b)^a \, mod \, (p)$$

Let us consider that two parties, 'Alice' and 'Bob' initiate a session and implement Diffie Hellman algorithm for their communication. They securely exchange the values of 'p', 'g'. The steps involved in the key generation of Diffie Hellman at each party is as follows:

- After both the parties agree to use two values 'p', 'g', Alice randomly selects a number 'Sa'. The 'Sa' is not revealed in public and is known only to Alice herself.
- Then she calculates the value of 'Ta' using the following formula
$$Ta = \ g^{Sa} \, mod \, p$$
- Alice then transmits 'Ta' to Bob.
- Bob, on the other hand, selects a random number 'Sb' and calculates 'Tb' using the following formula
$$Tb = \ g^{Sb} \, mod \, p$$
- Bob then transmits the calculated 'Tb' value to Alice.
- Upon reception of 'Ta' by Bob and 'Tb' by Alice, each of the parties generate the Secret key using the following formula
$$At \ Bob's \ End; \quad Key = \ Ta^{Sb} \, mod \, p$$
$$At \ Alice's \ End\text{:} \quad Key \ = \ Tb^{Sa} \, mod \, p$$
- Using the Secret Key obtained, the data is encrypted and/or decrypted.

Figure 5 shows Diffie Hellman in action.

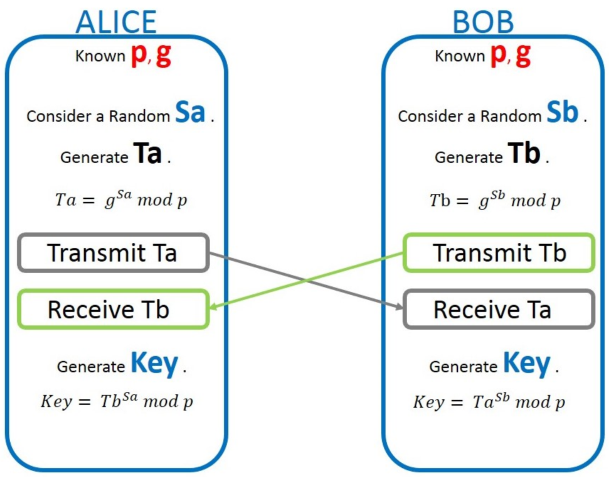


Figure 5 Diffie Hellman Algorithm [Adapted from 15].

In the entire method, the values 'Sa' and 'Sb' are known only to the respective entities. The values 'Ta' and 'Tb' are sent to the communicating parties. Thus, even if the hacker lays his hand on these values, he would not be able to generate the 'Key' as for the generation of the key, the 'Sa' or 'Sb' value is needed which is known only to the owner. If we consider, 'Rob' as the eavesdropper, then Figure 6 displays the knowledge of the values by each entity. The 'Red' colored ones are the known values and the values with 'Blue' color are not known by that particular party.

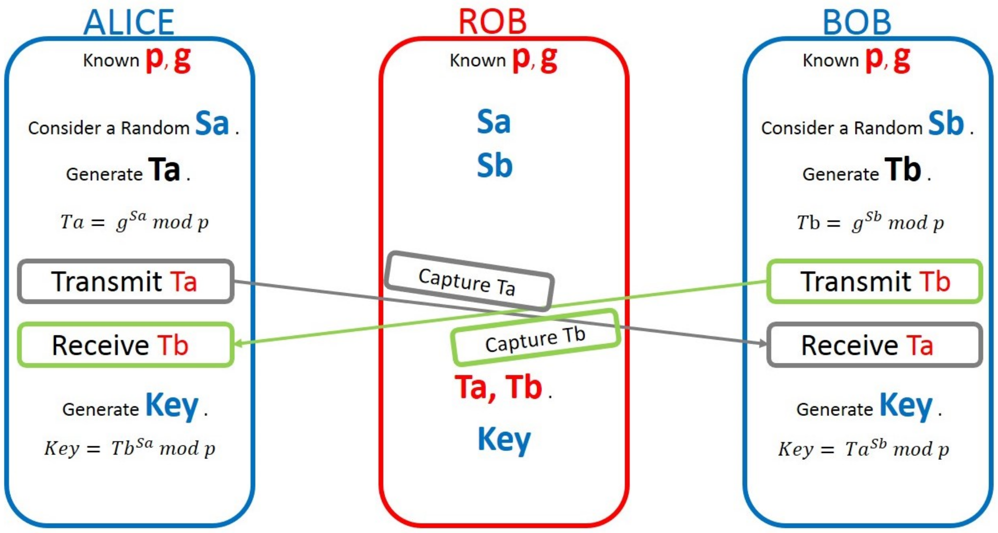


Figure 6 Known and unknown entities of Diffie Hellman amongst parties [Adapted from 15]

This algorithm is secure against Eavesdropping if 'g' is chosen properly and also if the group amongst whom the communication is going on, is large. Thus Diffie Hellman is used in public key encryption schemes such as ElGamal Encryption, Integrated Encryption Scheme etc. RSA in itself is motivated by the works of this algorithm. Protocols that yearn for forward secrecy, use Diffie Hellman since the key generation process is quick and these keys can be discarded after each session. [15]

2.5.1 MITM ATTACK ON DIFFIE HELLMAN

In [14] there is no provision of authenticating the parties involved in the communication. This leaves the entire algorithm vulnerable and susceptible to MITM attacks. Let us assume, ‘Alice’ and ‘Bob’ wants to communicate and there is an intruder ‘Rob’. The steps involved in the attack are as follows. Figure 7 is a pictorial representation of the following process.

- All the three are in possession of ‘p’ and ‘g’.
- On one hand, Alice selects a random number ‘Sa’ and generates ‘Ta’ using the formerly discussed formula. She then sends the ‘Ta’ to Bob. On the other hand, Bob selects a random number ‘Sb’ and generates ‘Tb’ and prepares himself to send it to Alice.
- ‘Rob’ captures the packet containing ‘Ta’. Rob then selects two random numbers ‘Sa1’ & ‘Sb1” and generates ‘Ta1’ & ‘Tb1’. He then sends ‘Tb1’ to Alice and ‘Ta1’ to Bob.
- Alice, upon reception of ‘Tb1’ generates the ‘Key1’ and Bob, upon reception of ‘Ta1’, responds with ‘Tb’.
- Rob is now in possession of both ‘Ta’, ‘Tb’ from Alice and Bob respectively. Rob then generates the ‘Key1’ for Alice’s communication and upon reception of ‘Tb’ from Bob, he generates ‘Key2’ for Bob’s communication.
- In the meanwhile, Bob generates the ‘Key2’ using the received ‘Ta1’ value.
- When the data transfer starts, Alice encrypts her message with ‘Key1’ and sends it out to the channel which is intercepted by Rob, who decrypts it with ‘Key1’ and replaces it with another message after encrypting it with ‘Key2’ and forwards it to Bob, who decrypts it with ‘Key2’.

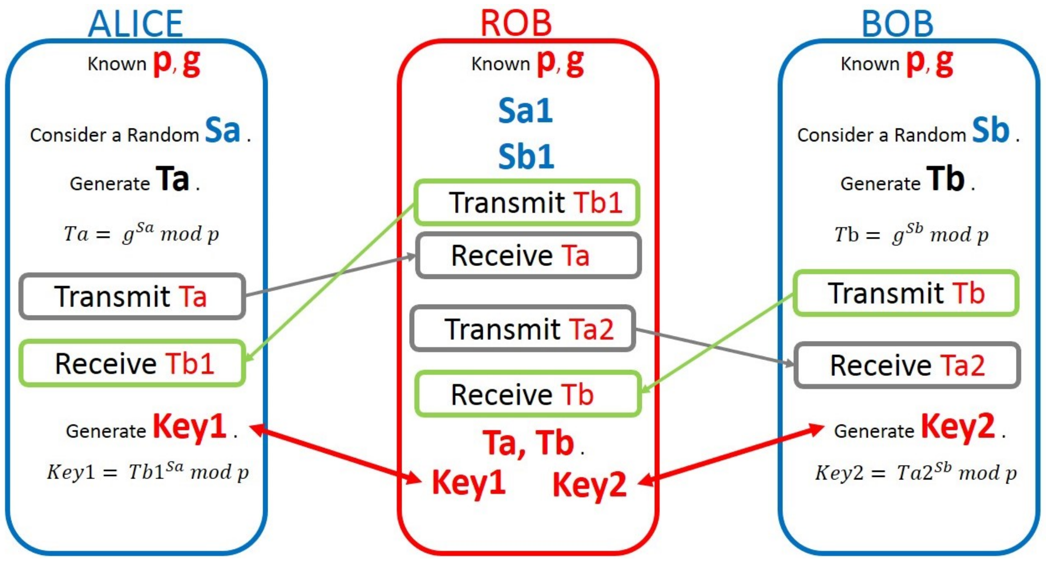


Figure 7 MITM in Diffie Hellman [Adapted from 15].

In this way, the entire session is hijacked by the intruder, who constantly reads each and every message that is transferred between Alice and Bob. Both Alice and Bob remain in a perception that they are communicating with each where in reality they are actually communicating with Rob [15].

2.8 Security Issues

Pawar et. al [61] and Anwar et.al [62] have presented the various security issue that the current existing network faces. They categorize the attacks under two broad categories: 'Passive' and 'Active'. 'Passive' attacks are the ones where the adversary intercepts the data transferred in the network, while 'Active' attacks are the ones where the adversary disrupts the normal functioning of the network by issuing some commands [57].

Under 'Passive' attacks, we have Eavesdropping, Traffic Analysis and Monitoring. In Eavesdropping, the adversary would aim to obtain the confidential data from the data packets that are available in the communication. In Monitoring, the adversary can read the secret or confidential information but cannot alter or modify it in any manner. In Traffic Analysis, the adversary makes an attempt to analyze the communication between the sender and the receiver. Herein the adversary cannot modify the data available in the communication channel but he can analyze the total amount of data being transferred and also analyze the communication path where he/she can later launch a Denial of Service (DoS). DoS implies to the type of attack which is launched at the Datalink layer, Network layer and/or Transport layer. This attack issues commands which inject fake or dummy packets into the channel thus disrupting the network facilities. Man In The Middle(MITM) is an attack wherein the adversary disguises himself as the other authentic party and initiates a communication between the two authentic parties [57] [58].

Under 'Active' attacks, we have Spoofing, Modification, Fabrication, Modification, Denial of Service, Wormhole, Sinkhole and Sybil attacks. In Spoofing attack, the malicious node presents itself as the other entity forcing the sender to alter his topology. In Modification attack, the adversary either alters the contents of the packet or modifies the routing route making the sender to transfer packets via a longer path causing the communication delay. In Wormhole or the Tunneling attack, the adversary obtains the packet and tunnels it to another malicious node wherein the sender perceives that he/she has found a shortest path in the network to its destination. In Fabrication attack, the attacker inserts counterfeit information into the data or generates false routing messages misleading the communicating parties. In Denial of Service attack, the network resources are consumed or clogged by sending dummy or fake packets into the communication channel. Sinkhole prevents the complete information to be obtained by the base station and Sybil is an attack wherein the malicious node connects with other malicious nodes via its secret key and thus numerous copies or instances of malicious nodes are ready for an attack [57].

2.7 SECURITY GOALS

As with any security research work, this work would be incomplete without the Security Goals analysis. Following is a short discussion on the Security goals which act as pillars to any Network Security Research work.

Confidentiality, Integrity and Authentication are the three major pillars of the network security. By Confidentiality, we refer to the scenario wherein if an illegitimate person eavesdrops into the traffic, the collected data packets would yield no information using which the eavesdropper could gain knowledge on the key or the data contained within the packet. By Integrity, we refer to the scenario wherein the message sent by the sender is exactly the same as the message received by the receiver. Both and/or either of the parties must identify the change made by any illegitimate person, to the received data packet. By Authentication, we refer to the scenario wherein the sender is validated & verified by the receiver and vice versa. Both the communicating parties must be able to identify that an illegitimate person is trying to enact as the other [63] [64].

The other security goals which also play their vital roles are Non-Repudiation, Forward Security, Backward Security, Access Control, Certification and Availability. Non-Repudiation implies to the situation where the sender cannot deny of the transaction it made later on. Forward Security implies to the scenario wherein an illegitimate person should not be able to deduce the keys which would be utilized for the upcoming communication based on the knowledge of the set of the previously used keys. Backward Security implies to the scenario wherein an illegitimate person should not be able to derive the keys which was utilized for encrypting the previous messages, based on the set of future keys. Access Control defines who can access what and when. It restrains the user to access certain features if the device finds the user to be illegitimate. Certification refers wherein a trusted third party authenticates the source. Finally, Availability is the scenario wherein only the legitimate users are allowed to use the system resources. In short, system resources are available only to the legitimate users [64] [63].

CHAPTER 3 RELATED WORK

Through this chapter, we make an attempt to present the works of various researchers in the fields which in some way or the other, has influenced this thesis. We begin with the studies and tests performed on the mobile devices and the sensors concerned to them. Then we give a glimpse of the studies performed towards key generation approaches, followed by the studies on Diffie Hellman algorithm. We also look into some of the techniques used to secure the message in the network with the help of biometrics and other related works. We finally conclude with the contribution of the studies and their limitations.

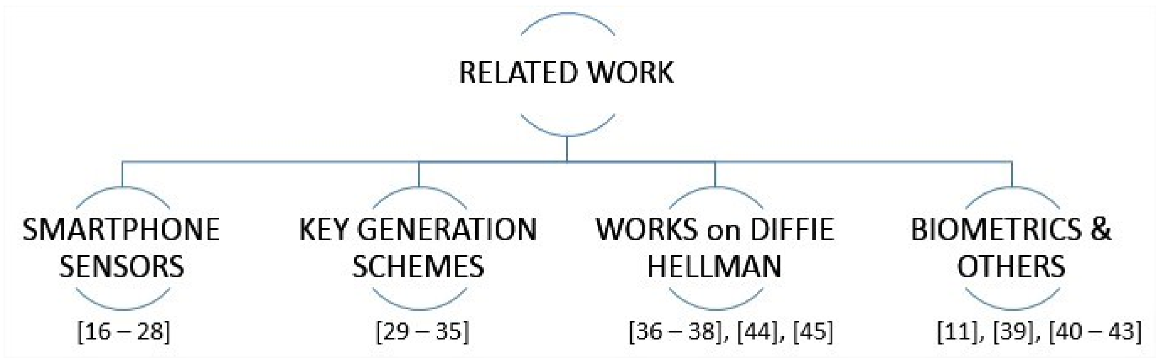


Figure 8 Overview of the Related works discussion

3.1 LITERATURE SURVEY ON SMARTPHONE DATA

Here we make an effort to survey through the works of various authors who have utilized smartphone and /or sensors for obtaining their objectives. This section is divided as illustrated in Figure 9.

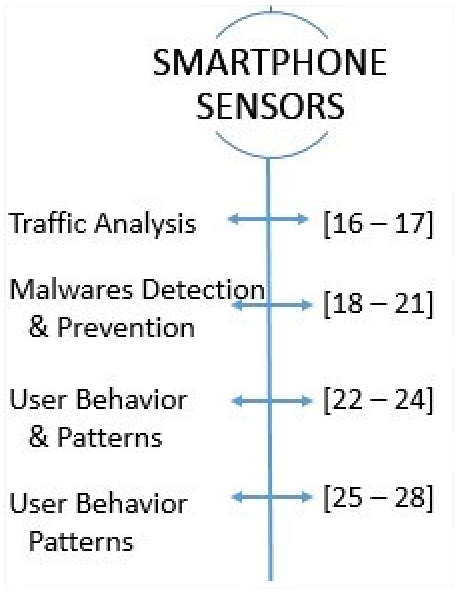


Figure 9 Survey on Smartphone Sensors

3.1.1 SMARTPHONE TRAFFIC ANALYSIS

The authors of [16] [17] [18] speak about the traffic dynamics generated from the smartphones. They collected the data from the network service providers and analyzed them. The devices list included smartphones which run on iOS and Android platforms, Ipods and personal computers. They distinguished between the devices based on the header information retrieved from the packets. The main objective of these studies were to help the network provider manage their resources efficiently based on the analyzed increase or decrease of the demand of the network.

Shafiq et. al [16] consider the traffic generated from machines such as asset trackers, security cameras, healthcare applications, traffic from smartphones etc. and presents the characteristics of the traffic in terms of temporal dynamics, device mobility, application usage and network performance. Maier et. al [17] analyzed the device usage from the network perspective by studying the packet level data of over 20,000 customers. They used Bro NIDS to anonymize, classify data and extract header from them. From their observation, they concluded that, DSL lines remained same in each trace but the count of the devices increased significantly. They also found that multimedia is the mostly found content in HTTP, iPhones & iPods are the most commonly used devices with Safari as mostly used browser. Li et. al [18] focused on understanding the user behaviors of three mobile platforms- Android, iOS and Windows. The data collected from the big cellular network company was analyzed from two aspects; Traffic dynamics and User applications. They identified the platform of the mobile devices based on the Type Allocation Code (TAC) in the HTTP signatures. They finally concluded that iOS generated maximum traffics than Android and Windows.

3.1.2 MALWARE DETECTION & PREVENTION

The authors of [19] [20] [21] [22] present their works on detection and containment of the malwares in smartphones. They collected the application data from the device and their methodology mainly focused on mapping the signatures and behavior of well-known malwares. They developed systems for collecting and analyzing the data. Monitoring, detecting the flow of data and then notifying the user is the common strategy. All the studies relied mostly on training data and proposed solution for detection and prevention of malwares in smartphone devices.

Liu et.al [19] used a Static and Dynamic analysis technique to detect malicious code. In static analysis, there is a Black list and a White list detection module which has the signatures of the malicious code. Regular expression and key-word matching module helps in identifying the malicious code. A decompiling module helps in decoding the application bytecode and then it is compared with the lists. If a match is found, then it alerts the user else it enters into Dynamic analysis mode where the android system output log is used to monitor the sensitive behavior. The collected log is filtered based on the sensitivity of information and then that particular log is analyzed for malicious activities. They have analyzed 400 applications with a detection rate of 78%. Louk et. al [20] developed 'MDTN- Monitoring, Detection, Tracking & Notification' system which monitors the application via signature behavior analysis, detects the malwares and reports them to the user. Their application constantly monitors the application activity such as access to registry, main memory etc. It detects malware using Anomaly based technique and Signature based technique. Upon detection, the system tracks for the source, starting with the detected taint or behavior and finally notifies the user.

Burguera et. al [21] proposed a framework for malware detection by collecting traces based on crowdsourcing. The framework detected malware based on the application behavior and matching it with the data collected from the crowd and another dataset which consist of the artificial malware. For collecting the behavior related data from the unlimited real users, an application called Crowdroid was developed which monitors Linux kernel system calls and send it to the centralized server for processing. Finickel et.al [22] presented a methodology for the analysis of application behavior based on the logs collected from the android logging system. The log entries are then mapped to bit vectors where each dimension is requested permission or action. These vectors are then fed to the Self-Organizing Maps which is a neural network learning system, to generate the pattern of requesting permission and performing actions of the app and then compare that with the malicious ones.

3.1.3 USER BEHAVIOR & PATTERNS

Authors in [23] [24] speak about the user behavior and patterns associated with it. In these studies, data is either given by the user or is collected from the logs. Different interaction techniques and the user behavior are analyzed using the sensor logs.

Hirabe et. al [23] gives an idea on logging all the touch operations, analyzing them and visualizing them by collecting them from the OS. The author claims that, using the touch patterns of the user, the user behavior can be analyzed and utilized for different purposes. These patterns include single/multi touch operations, pinch/zoom operations, pressure while touches and the average speed of the user while performing those touches. Tian et. al [24] proposes a technique which predicts the route without involving the user to feed in the input. GPS sensor data is used and Time Clustering Algorithm for classifying the destinations based on pre-stored daily destination behaviors and Trajectory Matching Algorithm for matching the current location to the stored locations in order to predict the destination is used. Turning Estimate Algorithm helps in storing the location at optimum times thus reducing the cost of energy required in processing and translating the co-ordinates. Linear Regression Algorithm is used to detect a turn and then store the location where the turn is made, as a straight streak is not required to store, it is usually a straight streak between the turns. Google Map Service is used to show the detailed route and co-ordinate to map location translation i.e. street name, city name etc.

3.1.4 USER BEHAVIOR PATTERNS

The works in [25] [26] [27] discuss about the user behavior patterns based on the data collected from various sensors available in the device. The User behavior includes video watching patterns, Mood assessment, Ontological user profile, motion type and battery consumption patterns.

Datta et. al [25] present four different techniques to preserve smartphone power consumption. Substitution, Suppression, Piggybacking and Adaptation are the four techniques which re-assess the need and utility of a particular resource of device and recalibrate its usage. A context aware application manages the application with the help of algorithms which predicts the next charging opportunity and battery lifetime. The automated power construction technique called PowerBooter makes use of sensors to record the battery discharge behavior and PowerTutor which displays the power consumption by each application. The Batterylogger stores the log of real user's usage time, power consumption, network usage and their patterns is the final module. Based on this, they state that the battery usage pattern varies from person to person. Ma et. al [26] addresses the mood related mental health problems by proposing MoodMiner, a framework for assessing and analyzing mood in daily life. This framework extracts human behavior by using mobile phone's sensor, communication data and assess daily mood. Mood is represented with the aid of three dimensions and four features. Displeasure, tiredness and tensity are the three dimensions while location, micro motion, communication frequency and activity are the four features. The location is collected using GPS and clustered using the K-means algorithm. Micormotion such as user picking up the phone for a few seconds and doing nothing useful, are extracted from the accelerometer raw data. Communication frequency are based on the call and text messages sent/received by the device. Activity includes walking, sitting, etc. which are also collected by the accelerometer data. Based on the previous mood and the accumulated features, the current mood is determined. Bedogni et. al [27] collected the data from the accelerometer and gyroscope sensors to detect the user's motion type. The authors collected the training data to classify the testing data into different motion types. Different classification algorithms were used to classify the training data into different motion types. Finally, the motion type recognition algorithm is integrated into the mobile application to detect the motion type.

Bo et. al [28] developed a mobile application named SilentSense, which authenticates the users without disturbing the user's operations. The application uses the touch behavior and other features such as motion, walking etc. features which can differentiate the owner of the device from the guest users. It silently senses the touch behavior and uses SVM to store the user touch pattern. For instance; if a guest is using an insensitive information bearing app such as a game app then security feature is off and if the guest is using a sensitive feature bearing app then security feature is turned on. Gait feature is also recorded in order to sense the discrepancies invoked when the actual owner is using the mobile. In short, SilentSense app uses the touch pattern, pressure for touches and the gait to authenticate the owner and thus assign the level of security to the information.

3.2 LITERATURE SURVEY ON VARIOUS KEY GENERATION METHODS

The authors of [29-35] discuss on the various ways keys have been generated using the Diffie Hellman approach in different fields such as VANETS, mixnets, Relay networks etc. They present us the techniques they have adopted to encounter MITM in DH algorithm, such as use of random number generator, messages, images etc.

Islam et. al [29] uses images as secret keys rather than using the traditional binary secret keys. Letters of the message to be transmitted are translated into their respective 8-bit codes. These codes are binary in nature. This 8-bit code is then searched in the image pixel values. The coordinates of the location on the image where the match is found, is retrieved and stored in a column layout rather than in (x, y) layout. This is sent as the cipher text. The receiver will scan the image and locate the pixel which match and then translate to the original message. They claim that it is highly secure due to the large key size and conclude that the performance of the proposed technique is better than AES, 3DES and DES.

Khader et. al [30] presented an overview of the MITM attack in DH protocol and then proposed a method to secure the Diffie Hellman. Their method involves the use of Geffe generator which generates binary sequences with high level of randomness. They conducted a series of tests such as frequency tests, serial test, poker test etc. to check the randomness of the generator's sequences. The generated sequences were used to calculate the private keys and the shared keys. Also they stated that 'p' should be a prime number and g should be primitive root of Modulo p. The private key is not sent into the channel and is saved as hashes in the server. Their proposed method also provided non-repudiation by identifying the parties from their user information.

Ahmad et. al [31] talks about the mix-servers that are used in a decryption mixnet, which continuously receive encrypted messages as input then decrypt and shuffle them to get a new output. From this output, the messages are regained. Here the authors present a decryption mixnet by comparing the Symmetric key encryption algorithm and the Asymmetric key encryption algorithm. All of that is attained via simulation. Both, symmetric and asymmetric key encryption algorithms are evaluated based on several criteria such as number of messages traversing through the mixnet, number of mix-servers involved and the key length. They have concluded stating that Enhanced Symmetric Key Encryption Based Mixnet (ESEBM) requires the least time as it uses optimal cryptographic technique for encryption and decryption purposes. Also while passing messages, it does not need large number of overhead for messages. RSA is slow due to the repeated encryption and it requires more time than the rest. ElGamal requires less time than RSA as the sender only performs a single encryption for all mix-servers. Thus for mixnets, ESEBM is the best followed by ElGamal and RSA.

Wang et. al [32] proposes a signcryption scheme based on the hardness of q-dDffie Hellman problems and proved that the semantic security under q-Diffie Hellman Inversion problem assumption with its unforgetability under q-Strong Diffie Hellman problem assumption. The proof of security is provided in random oracle model. They challenge the signcryption proposed by Libert and Quisquater whose operation has the same cost as an Elgamal encryption with reverse operation requiring only three exponentiations and one pairing to be done. The authors claim that their scheme is the most efficient among the existing schemes.

Sha et. al [33] propose a cloud supported protocol to enable secure data reading from the isolated smart meters. These were to be deployed to secure the sensitive data existing in the Smart grid systems where the smart meters collect the large amounts of energy used by the Smart grid systems. Most of the proposed authentication protocols which authenticate the smart meters in smart grids ignore to control the readings to the smart meters. In the proposed protocol by the authors, an Asymmetric key based authentication is designed for the reader-cloud authentication and then a new one-time symmetric key is generated by the meter reader which is shared by the smart meter. Then, authentication is done between the reader and meter based on the symmetric key. Hence the reader has to authenticate itself with each and every smart meter available in the grid. They claim that the proposed protocol is lightweight and secure.

Sampangi et.al [56] proposed a security suite for Wireless Body Area Networks where two key management schemes are presented. In the proposed schemes, the need for key exchanges is removed by enabling the sender and the receiver to generate keys independently. The authors make use of the Reference Frames concepts in order to acheive their goals. Although their scheme ensures the secrecy of the keys and is less prone to attacks, they have some limitations. The initial set of Reference Frames that are used are dummy values and are uploaded by a human which might be prone to social engineering or human errors. The next limitation is the loss of Acknowledgement frames i.e. there is no tracing to know whether the acknowledgement is lost or data has failed to reach the other end.

Khan et.al[34] proposed a symmetric key generation and pre-distribution scheme. The authors make use of a Symmetric matrix and generator matrix of maximum rank distance(MRD) codes. This scheme is applied on the wireless sensor networks which are resource constrained and hence key distribution became very difficult. In the proposed approach, sensor nodes are grouped and some information is stored in each node for the generation of the keys. The authors claim that their scheme reduces the communication overhead to setup a link key as it requires only two messages to establish a link key between any two nodes. They also claim that their scheme also provides the highest level of network scalability and connectivity. The division of sensors has isolated the node capturing effect to one specific group. The proposed scheme provides low memory usage, 100% network connectivity, scalability and low communication overhead without sacrificing authenticity, integrity and confidentiality aspects.

Fuloria et. al [35] discusses about the key management in electricity transmission and distribution for communication within substations and between substations. In electricity network, key management is a challenging task because of the expanse of the network and the resource constrained environments. The authors have presented a different variant of symmetric key and public key protocols which is simple, usable and cost effective. They presented a detailed threat model, analyzing a range of scenarios from physical intrusion to supply chain attacks. In conclusion, they claim that protecting the communication within the substation provides less benefits when compared to the use of cryptography to secure wider area communications between substation and network control center.

3.3 LITERATURE SURVEY ON SOME OF THE WORKS ON DIFFIE HELLMAN

Following are the works which have detected some of the ways through which MITM attacks are possible in the Diffie Hellman algorithm and some of the techniques using which the attack can be mitigated. We also discuss briefly on some other works which have been explored in Diffie Hellman algorithm.

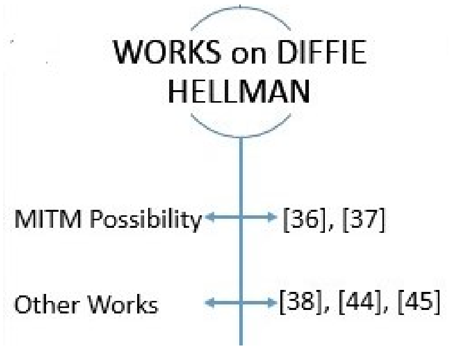


Figure 10 Works involving Diffie Hellman Algorithm

3.3.1 MITM ATTACK POSSIBILITY in DIFFIE HELLMAN

Kumar et. al [36] speak about the Diffie Hellman and how MITM is possible in Diffie Hellman protocol. They show how the key space can be exhausted by brute force. They suggest double encryption of the message along with an Identifier as a safety measure from the attacks. They also suggest to make use of Digital Signatures concept. They present Station to Station Protocol (STS) as example which employ RSA signature for getting public and private keys. The authors have neither implemented nor tested their proposed suggestions but mentioned it in their future work.

Van Oorschot et. al [37] state that in a Diffie Hellman key exchange, if the key exchange takes place in a certain mathematical environment, then the algorithm is vulnerable to specific MITM attacks and thus the keys are compromised. It also talks about Station to Station Protocol (STS) which is a three-pass variant of the Diffie Hellman protocol which allows the establishment of shared secret key. It also provides mutual entity authentication and mutual explicit key authentication between the two parties.

3.3.2 OTHER WORKS ON DIFFIE HELLMAN

Chang et. al [38] talks about the extension of Diffie Hellman to multiple party key distribution in a wireless relay networks instead of the traditional two party. Multi-party relay network is one where multiple parties communicate with each via a single relay. They proposed two efficient key exchange protocols and ran performance comparison with the existing ones. In group Diffie Hellman, the product of the individual private keys is a factor of obtaining the shared key. They explained the two proposed protocols and compared them. They conclude that their Diffie Hellman key distribution protocols for small group network setting is highly efficient and for large group network setting is practically efficient. The comparison suggested that the proposed protocols are not derivative variations of the existing ones. They also conclude stating that Protocol 1 is optimal when there are three entities in the network.

Mejri et. al [44] propose a secure variant of Diffie-Hellman algorithm for groups that are secured by a pre-shared key, against MITM attacks. Their proposed scheme can be used by several types of authentication and encryption VANET applications. They discuss a wide variety of problems in VANETs and use their proposed method to solve them. They secure the group against MTIM by calculating the group key as XOR bitwise between initial group key and pre shared secret.

Zhang et. al [45] implemented six kinds of optimization algorithm for Elliptical Curve Diffie Hellman algorithm for key agreement and pairwise key creation between sensors in wireless sensor networks comprising of IRIS nodes. The author then tests and compares the implemented algorithms w.r.t. RAM/ROM consumptions, initialization time and key establishment time. They presented their results and showed which settings and which order of execution would yield ECDH more effective.

Stulman et.al [68] proposed an algorithm, named as Spraying Diffie Hellman, to facilitate secure key exchange during conversation handshake on the secure channel in which the MANETs are built upon. Their algorithm requires no prior knowledge and user intervention as the exchange is done by using the fluctuation of the network topology inherited. They use the topology and the features such as IP address or IMEI number or phone number as key generation material. They also split the message to be transferred into segments and each piece is provided with a hash of the entire message. The match confirms the authenticity of the sender and also helps in detecting the MITM as the hash cannot be formed without the presence of all of the message segments. Also each segment is sent on a different route to the destination. They also claim that their algorithm also preserves forward security as they to change the key for each communication.

Shen et.al [69] propose a secure key agreement protocol in which two mobile devices can establish a shared secret without prior knowledge using Diffie Hellman with less communication and computation overhead. They use Random number generation for generating the key generation material and their XOR product for mutual authentication. It only after the authentication that the keys are generated. After confirming the authenticity of each entity, they also have to commit to the values they are going to use for the key generation. They claim that the possibility of an adversary to launch a MITM attack is close to 2^{-k} where 'k' is the number of bits present in the authentication value.

Yang et.al [71] propose a group key agreement protocol which reduces the time complexity from O(Nlog2N) to O(N) thus reducing the overlapping computation and data packet sending time. They make use of Binary trees in each node for generating the keys using the Diffie Hellman approach. Their technique functions in two phases, wherein the first phase involves the selection of the managing node whose responsibility is to gather and dispatch the information of generating group key and in the second phase is deal with the binary tree at each node independently. They claim that their technique could detect MITM as the node would not be able to decrypt the files as it is not in possession of the proper key as this key is generated based on the ingredients accumulated by the managing node from all the nodes available in the group.

3.4 LITERATURE SURVEY ON BIOMETRIC AND OTHER ALGORITHMS

This section discusses on how the biometrics are used for authentication and encryption techniques. It also discusses on some other related techniques and methods used for similar purposes.

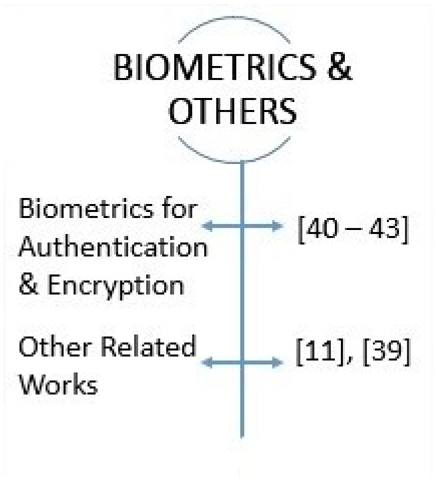


Figure 11 Other related Works

3.4.1 BIOMETRICS for AUTHENTICATION & ENCRYPTION

The authors of [40-43] speak about how they have utilized biometrics for authenticating and/or encrypting the data. Most of them have preferred fingerprint as the biometric. Some have explored different ways in which fingerprint can be used while others discuss the areas where it can be implemented.

Liang et. al [40] propose a fingerprint encryption scheme which is based on threshold. Their proposed technique is, unlike the Fuzzy Vault, which relies on templates or the encrypted templates in Fuzzy commitment based scheme while regeneration of the key which protects the biometric key in a polynomial. They claim that Fuzzy Vault has the risk of information leakage for the stored biometric template. Fuzzy commitment, on the other hand, stores an “encrypted” template but it has a short and unstable key. Dynamic biometric key generation stores neither the template nor the secrets and also it has a low number of effective bits. The proposed biometric cryptosystem, does not require to store the template and also the key is relatively long. But it has high time complexity, low Genius Acceptance Rate (GAR), a coarse quantitation and a pre-align work. Jain et. al [41] propose a multimedia content protection framework which is based on the biometric data of the users and a layered encryption/decryption scheme. They use a combination of symmetric and asymmetric key systems for the fulfillment of the scheme. The hardware features such as the hard drive serial number etc. are also used in the encryption/decryption processes. Watermarking and data hiding techniques could be utilized to address the requirements. They make use of the intra-class variation which talks about the differences in the extracted fingerprint's minutia etc. due to the sensor or image capture fluctuations. Sharma et. al [42] propose a framework that safeguard the sensor data in the cloud from unauthorized access and self-protect in case of breach. This data is the healthcare monitoring application data which is sent and stored in the cloud. The security is provided using biometrics and other related mathematical formulae and algorithms. They have used PKI for transferring the biometrics to the cloud before initiating the communication.

Wang et. al [43] proposed fingerprint based Biometric encryption. They have used Gabor filter ban to extract the fingerprint features and a binary key is bind with the filter bank. The extracted feature has fixed length and comprise of both local, global information. The binary key is bind to the filter-bank using traditional cryptographic algorithm such as AES, DES etc. The algorithm is evaluated on FVC database and the results show that Reference point localization has a great impact and robust reference point is important for this framework. Better fingerprint extraction algorithm and an efficient ridge texture enhancement algorithm for eliminating the texture noise could be used to improve the quality and performance of the algorithm.

3.4.2 OTHER RELATED WORKS

Suh et. al [11] present PUF designs that make use of the delay characteristics of wires and transistors that vary from chip to chip. They have discussed various different PUFs such as Ring PUF, Oscillator PUF etc. and their functionalities. The authors also explain how PUF enable low cost authentication and for volatile secret keys for cryptographic operations. These volatile keys can be used for both symmetric and asymmetric algorithms.

Quisquater et. al [39] explain about the Zero-Knowledge proof. In this, only the Prover must have the answers and only the Verifier should be able to verify the answers to his questions. The answers are not stored in the verifier's end. In this way, no one else except the prover can prove that he is legitimate and also minimizes the chances of replay attacks. Thus the chances of someone guessing the answers or the sequence of questions is astronomically small. Since none of the answers is stored in the verifier nor the prover, hence recording the packets would yield nothing useful. It is entirely based on the fact/assumption that the prover possesses some secret information using which it can prove itself to the verifier. Neither the secret information is stored in the verifier end nor is transmitted over the wire. Hence as mentioned before, recording the packets and observing the sequence would do no good at all.

3.5 Summary

In summary, the authors of [15-17] present us with the amount of traffic being generated by the mobile devices for resource allocation. The works of [18-21] detect malwares in smartphone devices using various different techniques. [22-27] focus on the User behavior among which [22-26] extract the pattern of the user behavior and interactions with the device but [27] authenticate the user based on the interaction of the user with the device and the gait as well. [28] continuously tracked the user behavior in the background to authenticate the owner from the guest. Hence this shows us that the sensors have been put into use for various different purposes and added to that there is tremendous amounts of traffic originating to and fro the device upon which the proposed algorithm can be applied to enhance the security. [11,39] are some new instances for authenticating and key generation. [40-43] present with works that have dealt with biometrics for key generation and/or authenticating the user and securing the data. [29-38,44,45,68-70] have encountered the MITM in DH and also have modified it for key generation. In most of the works, they have attempted to make the key as unpredictable as possible in order to prevent the possibility of MITM in Diffie Hellman.

3.6 Motivation and Research Objectives

From what we have explored so far, we noted that researchers are making an attempt to prevent and/or detect MITM attacks in Diffie Hellman by making the shared key content as random, as unpredictable and as unique as possible with the preservation of forward secrecy. The keys are generated either out of the hardware of the device which is unique only to that specific device such as IMEI or out of the values that the device generates such as its readings. These values tend to be random and unpredictable for an adversary if he intends to launch a MITM attack. Apart from that, the researchers have also made an attempt to minimize the transfer of key generation material for establishing a shared key.
As discussed in Section 3.5, researchers have attempted to make the keys unpredictable so that an adversary may not be able to initiate an unauthentic communication with either of the parties. Added to that, we have also observed how the sensors are put in use for various purposes such as malware detection and user behavior analytics proving that the values obtained from them are sensitive to movements and variations occurring with the device. Some of them have either transferred the key generation material out in the communication channel or have not acknowledged the reception of the data. Thus with the motive to eliminate the need for a separate transfer of the key generation material such as the ‘p’ and ‘g’ values out in the public and to render the keys as unpredictable as possible, we have developed a new scheme which would make use of the sensor data available at our disposal. None of the schemes, to the best of our knowledge have used the dynamically fluctuating aspect of the sensors as the key generation material towards securing Diffie Hellman from MITM attacks.
To address the research gap, the objective was to develop a new scheme which;

i. Avail independent generation of the keys
ii. Increase the unpredictability and randomness nature of the keys generated
iii. Facilitate two way acknowledgments to ensure a successful transfer
iv. Eliminate the need for a separate transfer of the key generation material such as ‘p’ and ‘g’

v. Satisfy the security goals such as Confidentiality, Integrity and Authentication along with additional security goals such as Access Control, Forward security, Backward Security and Non Repudiation.

vi. Provide security from security attacks such as modification, replay attacks, fabrication, invasion and most importantly, MITM.

CHAPTER 4 METHODOLOGY

4.1 PROPOSED APPROACH

The proposed approach involves two phases, 'Registration' phase and 'Data Transfer' phase. Although the 'Registration' phase occurs only once in the entire lifetime of the application, yet it plays a vital role in the entire algorithm. The entire approach is designed around the client-server architecture with the client being any device which has sensors & internet access and the server is believed to be in a remote location.

In the Registration phase, the application gathers the hashed fingerprint of the user, the PUF of the device and a set of Reference Frames containing the values accumulated by the sensors. The application then sends them to the remote server using the RSA algorithm for encryption over PKI. Once the application sends out its second acknowledgment to the server, it deletes the hash of fingerprint it collected earlier and safeguards the application along with its data in case of theft or loss of the device.

In the Data Transfer phase, the application and the server utilizes and/or updates the data it received during the Registration phase to attain the objective of this research. The application uses the combination of the fingerprint and PUF for authenticating the user, hence facilitating access control. Also once authenticated, it then generates the terms such as p, g, Sa etc. required for Diffie Hellman Key generation. Similarly, the server follows the same set of steps to generate its own terms and upon reception of the Ta from the client, generates the key. The server then sends the terms it generated to the client using which its key is generated. The client then accumulates new sensor values and updates the Reference Frames which are in its possession. Then the new set of terms are generated again and with the data which is to be sent to the server, is then encrypted using the key and sent to the server. The server then decrypts the received message and updates its own reference frames and acknowledges with the terms as before for the generation of the key. Upon reception of the second acknowledgement from the client, the keys are destroyed by both parties and the reference frames along with the key are updated. The values which act as the acknowledgment are the values stored in any random field from the Reference frame.

In short, at both the ends of the tunnel, the terms p, g, Acknowledgments, are the values contained in the fields of the Reference Frames. These fields are selected at random so that the prediction of the values utilized as the terms which are used in key generation, is made difficult.

The step by step detailing of the proposed approach is explained in the following subsection.

4.2 ARCHITECTURE OF THE PROPOSED APPROACH

This section discusses the architecture of each phase in the proposed approach as mentioned in the Section 4.1. The symbol '♣' represents the steps which are performed at the client application end and the symbol 'Ξ' represents the steps which are performed at the server.

The only assumption made in this thesis work is, during the Registration phase, the server and the Third parties involved in the PKI are secure for this one-time phase.

Figure 12 to Figure 15 displays the timing diagram of the proposed algorithm's Registration Phase.

♣ The application gets the fingerprint 'fp' of the user. Then it generates a hash of the obtained fingerprint 'h(fp)'. Once the hash is generated, the actual fingerprint is deleted.

♣ The application then obtains the PUF of the device 'PUF(phone)' and stores a copy of it in the device.

♣ Then, it performs a XOR operation between the hashed fingerprint & the PUF value. The outcome of XOR operation is then hashed again and is stored ('z' in our case). When the user accesses the application again, the application requests for his fingerprint and performs the above set of operations to generate the ‘z’ value. This is then matched with the stored ‘z’ value. Access to the features of the application is provided only if a match is found else access is denied. In this way, the ‘z’ variable helps in providing access control for the application.

- ♣ After the generation of ‘z’, the application then gathers the required Reference Frames (minimum of 5). The Frame structure is presented in Figure 16. The first 5 fields in the Reference Frame are the values collected by the sensors available in the device. This is followed by the Timestamp pertaining to the creation of the frame and the sequence number. The Sequence number is placed to prevent the replay attacks on either of the parties.
- ♣ The hashed value of the fingerprint ‘h(fp)’, PUF of the device ‘PUF(phone)’ and the Reference Frames, constitutes the message ‘m’ which will be sent to the server in this phase.
- ♣ Then the Message Digest ‘MD’ is generated which is a simple hashing of the message.
- ♣ This Message Digest is then signed with the public key of the device ‘pks’ to form the digital signature ‘CMD’. This preserves the Integrity of the message.
- ♣ This digital signature ‘CMD’ along with the actual message ‘m’ is encrypted with the public key of the server ‘pkSr’ and transmitted to the server over the communication channel.
- Ξ The Server, upon reception of the message, decrypts using its private key ‘skSs’ and extracts ‘m’, ‘CMD’ components from it.
- Ξ It then computes the hash of the message portion ‘h(m)’ and encrypts it with the public key of the client ‘pks’ to get the message digest ‘MD’ and verifies it with the received ‘CMD’.
- Ξ If it fails to match, the packet is dropped else the PUF of the device ‘PUF(phone)’, ‘z’ value and the Reference Frames are extracted from ‘m’ and stored.
- Ξ Now, the first two digits from the public key value is taken and is used as pointer to a particular field amongst one of the reference frames. For instance, if the key value is 25689 then the first two digits constitutes of 2 & 5 thus pointing to the 2nd Reference Frame and 5th Field. The value stored in the location specified by the

pointers, is retrieved and used for the generation of the Acknowledgment of the server.

- Ξ The retrieved value is taken and a Keyed hash operation is performed on it. The key used in the hashing are the first two digits considered in the former step where each value is incremented by one i.e. since the first two digits were 2 and 5, hence the key is (2+1)(5+1) i.e. 36.
- Ξ The outcome of the previous step acts as the Acknowledgment of the Server 'ACKs', which is encrypted with the public key of the client 'pks' and sent out to the client.
- ♣ The client receives the encrypted message, decrypts using its private key 'sks' and extracts the Acknowledgment 'ACKs'.
- ♣ Retrieves the value from the corresponding field in the reference frame using the same technique as mentioned earlier for the generation of server's acknowledgment.
- ♣ Then it hashes the extracted value ad verifies it with the received Acknowledgment from the server i.e. 'ACKs'. If a match is found, then the client uses the public key of the server to get the pointer using which it retrieves the value from the respective field and reference frame. It follows the same set of steps as described for the generation of the Acknowledgment of the server in order to generate an Acknowledgment for the client 'ACKph'.
- ♣ The Acknowledgment for the client 'ACKph' is then encrypted with the public key of the server and forwarded to the remote server.
- Ξ The server receives the encrypted acknowledgment, decrypts it and verifies it to ensure the completion of a successful transfer.

Since the entire timing diagram does not fit in the page, we have split the timing diagram into parts and represented in the figures that follow. Figures 12 to 14 are the parts of the timing diagram and Figure 15 display the flow of the timing diagram.

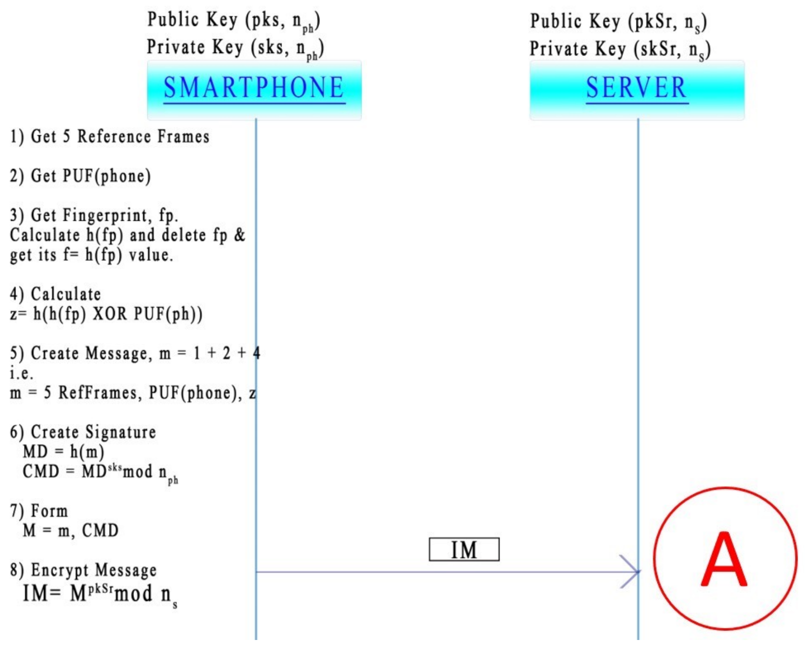

Public Key (pks, n_{ph})
Private Key (sks, n_{ph})
SMARTPHONE
Public Key (pkSr, n_S)
Private Key (skSr, n_S)
SERVER
1) Get 5 Reference Frames
2) Get PUF(phone)
3) Get Fingerprint, fp. Calculate h(fp) and delete fp & get its f= h(fp) value.
4) Calculate z= h(h(fp) XOR PUF(ph))
5) Create Message, m = 1 + 2 + 4 i.e. m = 5 RefFrames, PUF(phone), z
6) Create Signature MD = h(m) CMD = MD^{sks}mod n_{ph}
7) Form M = m, CMD
8) Encrypt Message IM= M^{pkSr}mod n_s
IM
A

Figure 12

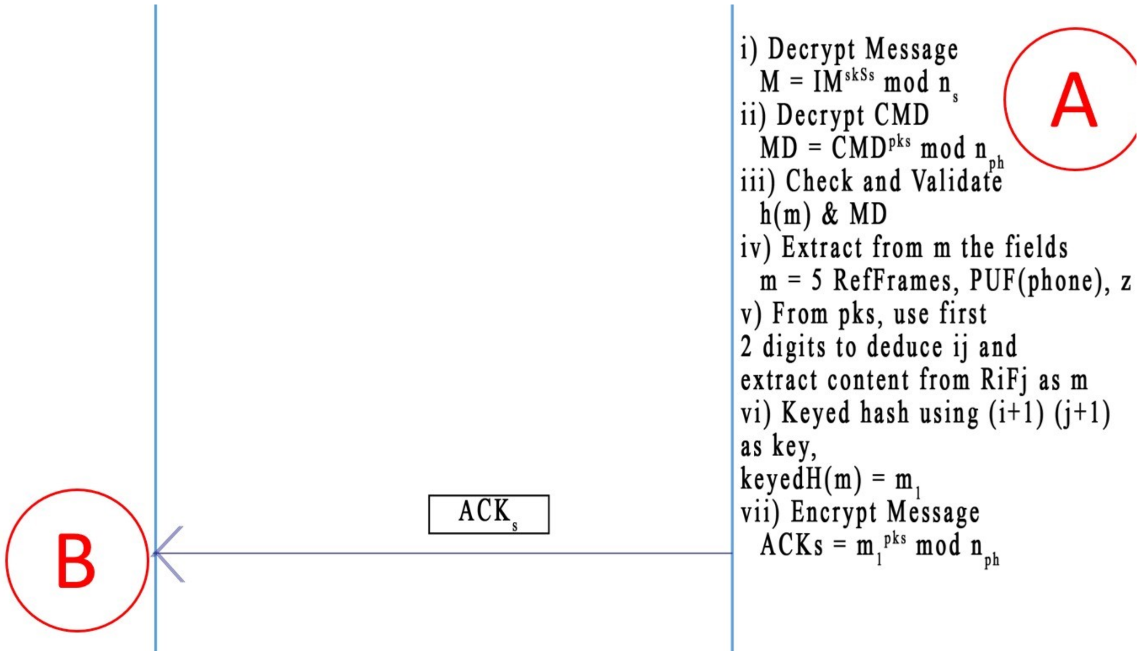

i) Decrypt Message M = IM^{skSs} mod n_s
ii) Decrypt CMD MD = CMD^{pks} mod n_{ph}
iii) Check and Validate h(m) & MD
iv) Extract from m the fields m = 5 RefFrames, PUF(phone), z
v) From pks, use first 2 digits to deduce ij and extract content from RiFj as m
vi) Keyed hash using (i+1) (j+1) as key, keyedH(m) = m_1
vii) Encrypt Message ACKs = ${m_1}^{pks}$ mod n_{ph}
ACK_s
A
B

Figure 13

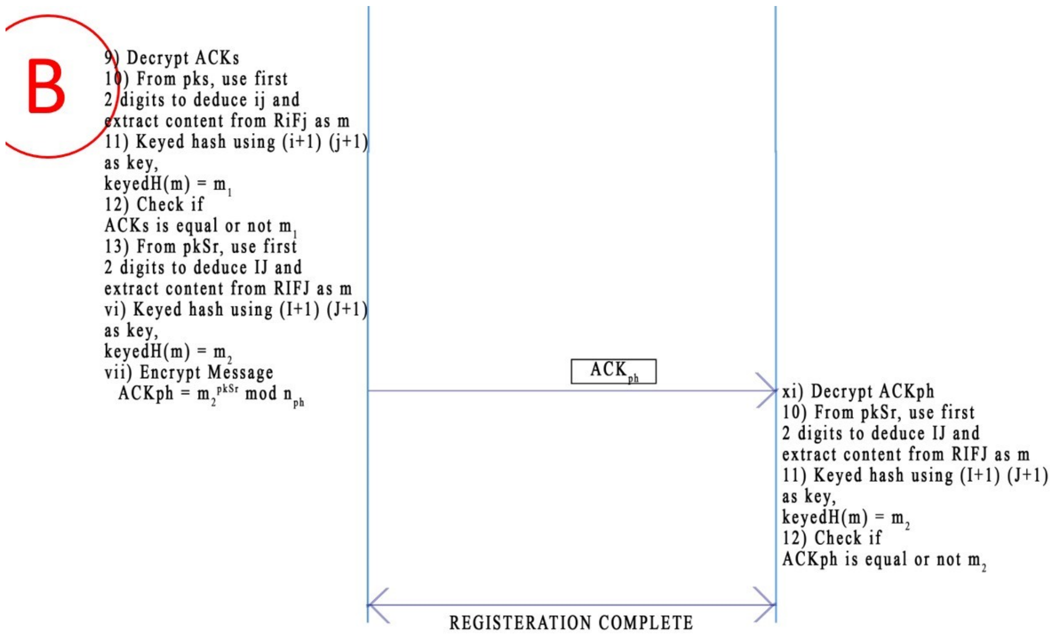


Figure 14

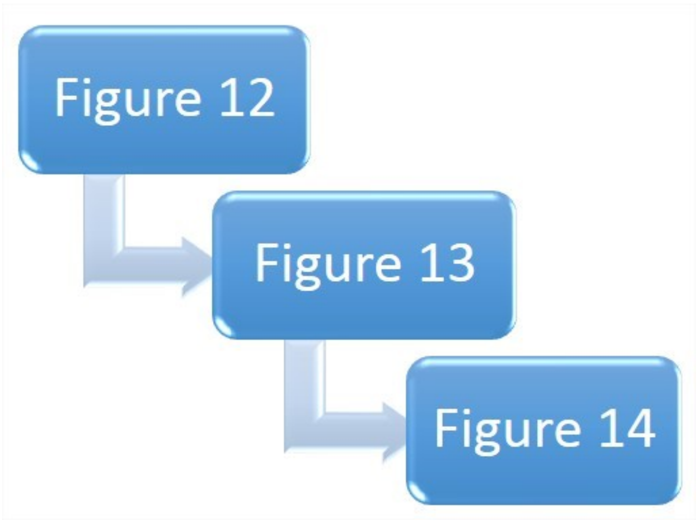


Figure 15 Registration Phase

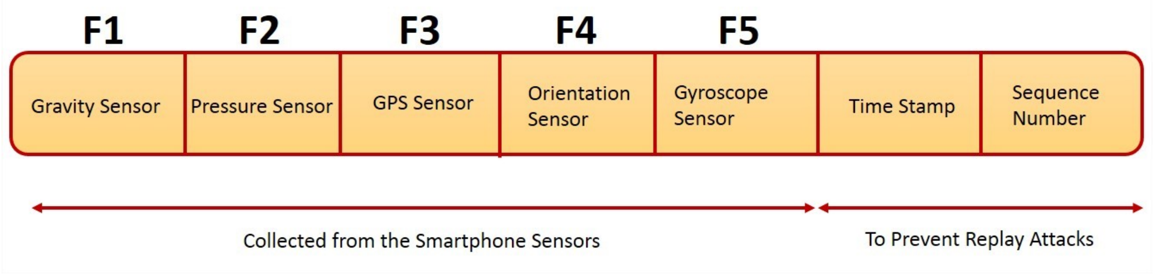


Figure 16 Reference Frame Structure

This marks the end of Registration Phase. The following is the description of the Data Transfer Phase. The assumption which was made during the Registration Phase is not needed here. The Data Transfer Phase is a repetitive one, which repeats itself for every required transfer of data. It is here that the objective of the research work is obtained.

Figures 17 until 24 displays the timing diagram of the proposed algorithm's Data Transfer Phase.

- ♣ The application gets the fingerprint 'fp' of the user. Then it computes the 'z' value using the stored PUF and matches it with the actual value of 'z' which was stored after the Registration Phase. Access is permitted only if a match is found else the user is restricted from accessing the application and its features.
- ♣ If the user is found authentic, the application then generates a Nonce 'Nc' with the help of first two digits from the stored PUF and uses it to point to a random field of a random frame. The value stored at that particular location, goes as Nonce 'Nc'. The entire fetching technique is similar to the technique used for getting the value for the Acknowledgments during the Registration Phase.
- ♣ Then a XOR operation is performed between the generated Nonce 'Nc' and the 'z' value. The outcome is then fed into the pseudo random number generator as seed. The digits of the random number thus generated are then used as pointers i.e. the first set of two digits are used to point to the location whose content would be used as the value for 'p'; the next set of two digits for 'g'; the next set of two digits for Acknowledgment of the phone 'ACKph' and the next set of two digits go for the Acknowledgment of the server 'ACKs'. Thus if 2365487596849 is the generated pseudo random number, then the content of R2F3 goes as 'p', R6F5 goes as 'g', R4F8 goes as 'ACKph' and R7F5 goes as 'ACKs' where RnFm is the mth Field in the nth Reference Frame.
- ♣ A random number is taken as the value of 'Sa' and 'Ta' is computed using the formula;

$$Ta = g^{Sa} \, mod \, p$$

- ♣ Then a simple hash of the Nonce 'Nc' is performed 'h(Nc)'. This hashed Nc along with the generated 'Ta', is encrypted using the first two digits' value, which was used for the generation of Nonce, as Key. This encrypted message ‘Tm’ is sent to the server over the channel.

Ξ The server receives the message ‘Tm’ and decrypts it using the key it obtained after the generation of Nonce ‘Nc’ with the help of the PUF it stored after the Registration Phase.

Ξ Once decrypted, it compares whether or not the values of the hashed Nonce it generated is same as the one present in the received message. If they are not, then the packet is dropped else the server generates the same set of values as the client’s application using the same procedure.

Ξ Then the server considers a random number as ‘Sb’ and computes ‘Tb’ using the formula;

$$Tb = g^{Sb} \, mod \, p$$

Ξ Now it uses the ‘Ta’ value it received in the message and ‘Sb’ value it possesses to generate the Secret Key with which it shall be decrypting the data it would receive in the successive steps. The formula for the Secret Key generation is as follows;

$$Secret \; Key = Ta^{Sb} \, mod \, p$$

Ξ Then the server performs XOR operation between the ACKs and Ta. The result of the XOR operation is then key hashed. Here, the values which were used as pointers to the location whose content was the ACKs is taken and is incremented by a value of one. This incremented value acts as the Key which is used for the keyed hashing. For instance, in the example we considered earlier, the content of R7F5 is used as ACKs. Thus the key in this case would be (7+1)(5+1) i.e. 86. The outcome of the keyed hash is stores as ‘c1’.

Ξ Then another XOR operation is performed between the Acknowledgment of the phone and the Secret Key generated earlier. Using a similar technique as discussed in the former step, a keyed hash operation is performed on the outcome of the XOR operation. The result of the keyed hash is stored as ‘s2’ in the server. This ‘s2’ is later on used to authenticate the message which the server receives from the client.

Ξ Upon completion of the keyed hash operations, the value ‘Tb’ and the value ‘c1’ are encrypted using the last two digits of the stored ‘PUF(phone)’ value as key. This forms the first Acknowledgment from the server ‘ACKn’.

♣ The client receives the 'ACKn' and it decrypts it using the last two digits of the stored 'PUF(phone)' value as key.

♣ Then it computes its own 'c1' following the exact same set of steps as that of the server, and verifies it with the received one. If it fails to match, then the packet is discarded else 'Tb' value is retrieved and Secret Key is generated using the formula;

$$Secret\ Key = Tb^{Sa}\ mod\ p$$

♣ Then the application performs XOR operation between the ACKph and Tb. The result of the XOR operation is then key hashed. Here, the values which were used as pointers to the location whose content was the ACKph is taken and is incremented by a value of one. This incremented value acts as the Key which is used for the keyed hashing. For instance, in the example we considered earlier, the content of R4F8 is used as ACKs. Thus the key in this case would be (4+1)(8+1) i.e. 59. The outcome of the keyed hash is stored as 'ACKm'

♣ The 'ACKm' is then encrypted using the same set of keys which are used for the Keyed hash process and is then forwarded to the server.

Ξ Since the server is also in possession of the values and pointers which are used to create 'ACKm,' it generates its own set and verifies them with the ones present in the 'ACKm' after decrypting the received message.

♣ The application then encrypts the data with the Secret Key 'E(D)'.

♣ Then it performs a XOR operation between 'ACKph' and the 'Secret Key'. The result yielded is then key hashed using the incremented values of the pointer terms used for 'ACKph' deduction. The outcome is used for authenticating itself to the server and is thus stored as 'SAt'.

♣ Then the random number which was generated during the previous Secret Key generation process, is fed as the seed to the random number generator to generate another random number. This new random number is then used to accumulate the values for 'p', 'g', 'ACKs' and so on which were accumulated in the previous cycle. The former set of values for 'p', 'g', etc. are stored as backup until it verifies the 'ACKn' which it receives later as response from the server.

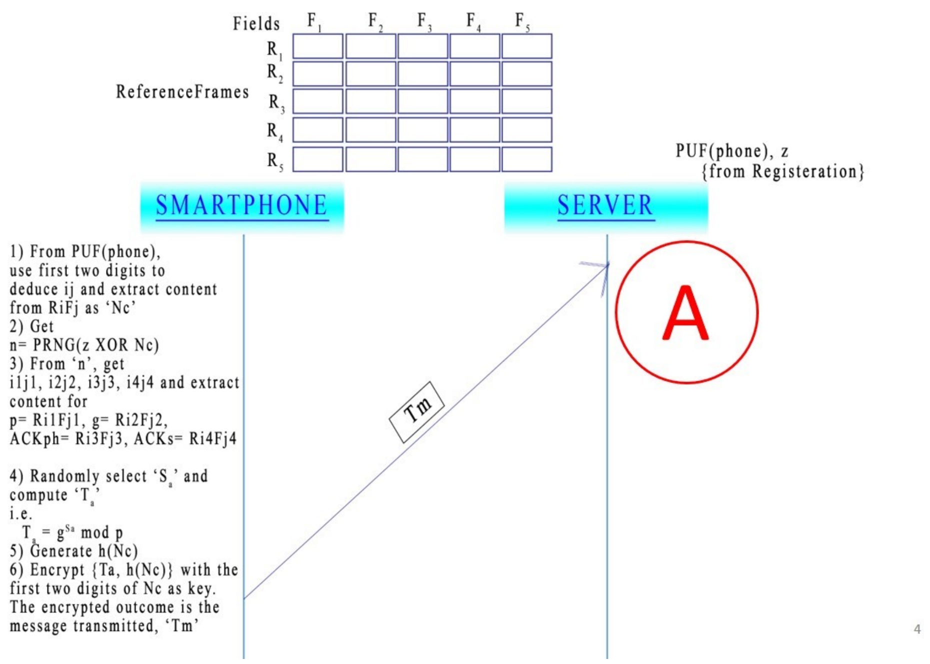
♣ Using the new values of 'p', 'g' etc., a new 'Ta' is generated using the formula.

- ♣ This new ‘Ta’ along with the calculated ‘SAt’, encrypted data ‘E(D)’ and sequence number ‘SeqNo’ is composed as a message, encrypted again with the Secret Key and is then sent to the server ‘TF’.
- Ξ The server upon reception of the message from the client application, decrypts it with the Secret Key and retrieves ‘SAt’ and verifies it with the ‘s2’ value which was stored earlier. If a match is found the Data is decrypted else the packet is dropped. After decryption, from the data, the reference frames are updated with new set of values contained in the actual data or as a part of the actual data.
- Ξ After decryption, the ‘Ta’ value is retrieved. Then it performs the same set of actions performed at the client end to generate and accumulate the new set of values for terms ‘p’, ‘g’, ‘ACKs’ etc., and new ‘Tb’ is generated using the formula. The former set of values for ‘p’, ‘g’, etc. are stored as backup until it verifies the ‘ACKm’ which it receives later as response from the client application.
- Ξ Now using the retrieved ‘Ta’ and the newly generated ‘Sb’, a new secret key is calculated again and stored which will be used for decrypting the data in the next cycle.
- Ξ Then the ‘s2’, ‘c1’, ‘ACKn’ are generated using the same approach as in the previous cycle. The ‘ACKn’ thus generated, is encrypted using the old secret key and is sent to the client.
- ♣ The client, upon reception of the ‘ACKn’, verifies it with the ‘c1’ it generated. If a match is found, it generates ‘ACKm’ using the same procedure s followed in the previous cycle. The generated ‘ACKm’ is then encrypted using the previous Secret Key and sent to the server. After transmission, the previous set of ‘n’, ‘p’, ‘g’, ‘ACKph’, ‘Secret Key’ etc. are deleted.
- Ξ The server upon reception of the ‘ACKm’, decrypts it and verifies it with the values it calculated using the same set of steps. If a match is not found then the packet is dropped but if a match is found, then the old set of ‘n’, ‘p’, ‘g’, ‘ACKs’ etc. are deleted.
- ✚ The above mentioned steps are continued until all the data has been transferred. After the last packet is sent,

- ♣ The client sends the 'c1' value it generated for the last data packet along with the sequence number to the server as 'ACKm'.
- Ξ The server upon reception of the 'ACKm' verifies it with the 'c1' it obtained after calculation and the sequence number 'SeqNo' it has for the previous data. If a match is found, then it performs a hash operation on the Secret Key it used in the previous cycle along with the Nonce 'Nc' value it has. The outcome of the hash operation is the last acknowledgment which is sent from the server 'ACK'. After the transmission of the 'ACK', all the terms used such as 'n', 'p', 'g', 'ACKph' etc. are deleted.
- ♣ The client application upon receiving the 'ACK' verifies it with the outcome of the hashing of the Secret Key used in the encryption of the last data and the stored Nonce 'Nc'. If a match is found, all the terms used such as 'n', 'p', 'g', etc. are deleted and the connection is closed.

This marks the end of DataTransfer phase. By the end of this, only the PUF of the phone 'PUF(phone)', 'z' value and the last set of updated Reference Frames are left in both the parties i.e. at the Server and in the client application. The rest are deleted i.e. the Secret Key used, the value 'n' which is used to accumulate 'p', 'g', 'ACKph', 'ACKs', 'ACK' etc. are deleted. This deletion prevents the access to those values and keys in case of theft of the device or in case of breach in the server.

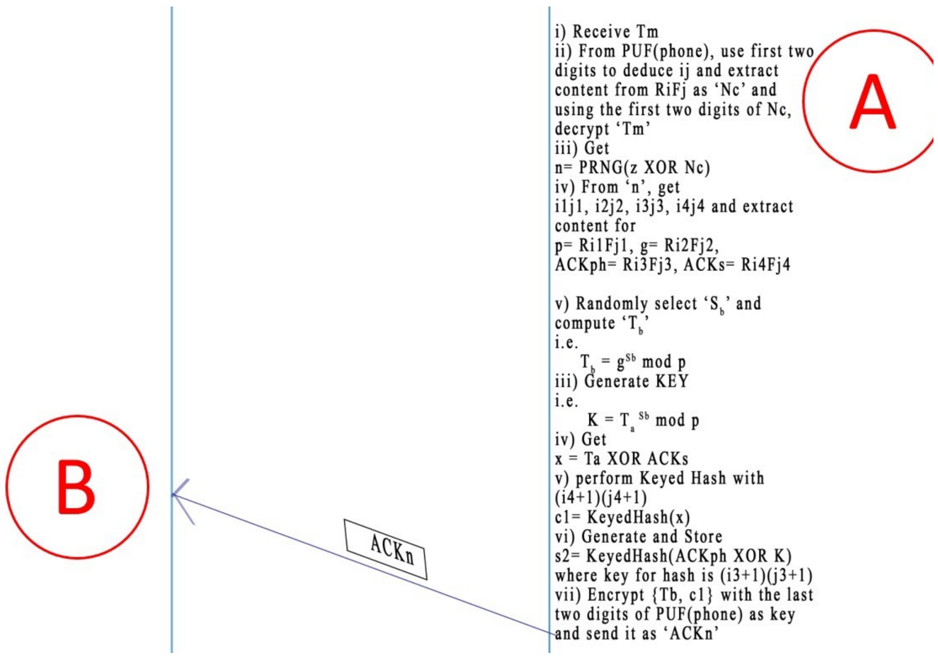
Since the entire timing diagram does not fit in the page, we have split the timing diagram into parts and represented in the figures that follow. Figures 17 to 23 are the parts of the timing diagram and Figure 24 display the flow of the timing diagram.



1) From PUF(phone), use first two digits to deduce ij and extract content from RiFj as ‘Nc’
2) Get n= PRNG(z XOR Nc)
3) From ‘n’, get i1j1, i2j2, i3j3, i4j4 and extract content for p= Ri1Fj1, g= Ri2Fj2, ACKph= Ri3Fj3, ACKs= Ri4Fj4

4) Randomly select ‘S_a’ and compute ‘T_a’ i.e.
$T_a = g^{Sa} \bmod p$
5) Generate h(Nc)
6) Encrypt {Ta, h(Nc)} with the first two digits of Nc as key. The encrypted outcome is the message transmitted, ‘Tm’

Figure 17



i) Receive Tm
ii) From PUF(phone), use first two digits to deduce ij and extract content from RiFj as ‘Nc’ and using the first two digits of Nc, decrypt ‘Tm’
iii) Get n= PRNG(z XOR Nc)
iv) From ‘n’, get i1j1, i2j2, i3j3, i4j4 and extract content for p= Ri1Fj1, g= Ri2Fj2, ACKph= Ri3Fj3, ACKs= Ri4Fj4

v) Randomly select ‘S_b’ and compute ‘T_b’ i.e.
$T_b = g^{Sb} \bmod p$
iii) Generate KEY i.e.
$K = T_a^{Sb} \bmod p$
iv) Get x = Ta XOR ACKs
v) perform Keyed Hash with (i4+1)(j4+1)
c1= KeyedHash(x)
vi) Generate and Store s2= KeyedHash(ACKph XOR K) where key for hash is (i3+1)(j3+1)
vii) Encrypt {Tb, c1} with the last two digits of PUF(phone) as key and send it as ‘ACKn’

Figure 18

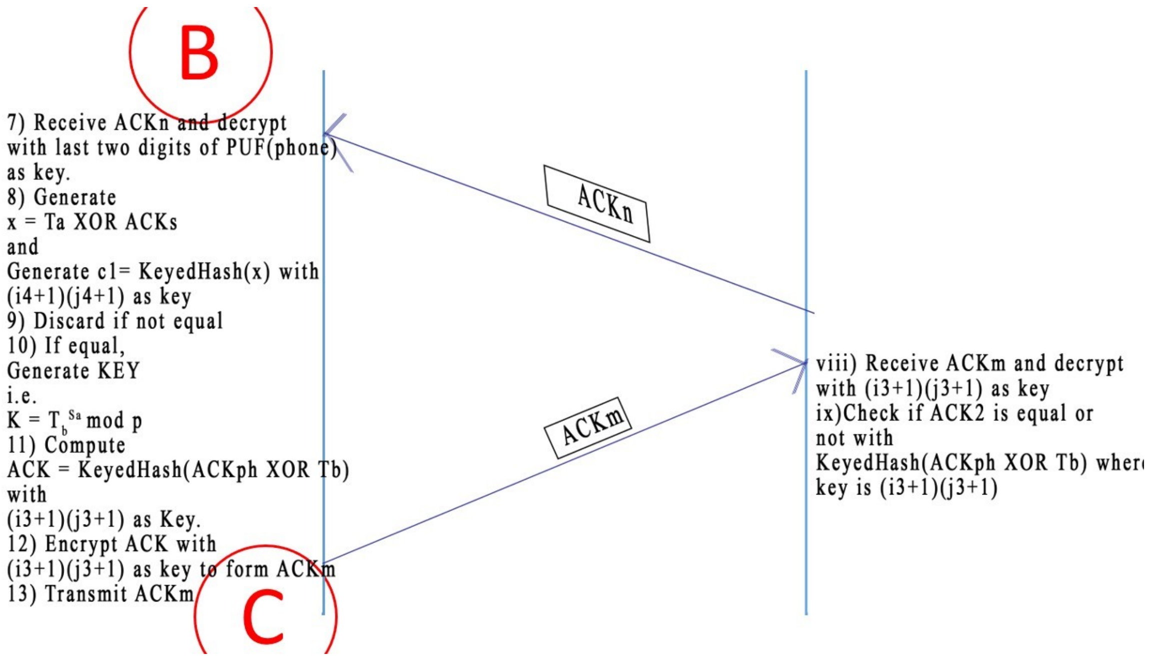

B
7) Receive ACKn and decrypt with last two digits of PUF(phone) as key.
8) Generate x = Ta XOR ACKs and Generate c1= KeyedHash(x) with (i4+1)(j4+1) as key
9) Discard if not equal
10) If equal, Generate KEY i.e. K = T_b^{Sa} mod p
11) Compute ACK = KeyedHash(ACKph XOR Tb) with (i3+1)(j3+1) as Key.
12) Encrypt ACK with (i3+1)(j3+1) as key to form ACKm
13) Transmit ACKm
C
ACKn
ACKm
viii) Receive ACKm and decrypt with (i3+1)(j3+1) as key
ix)Check if ACK2 is equal or not with KeyedHash(ACKph XOR Tb) wher
key is (i3+1)(j3+1)

Figure 19

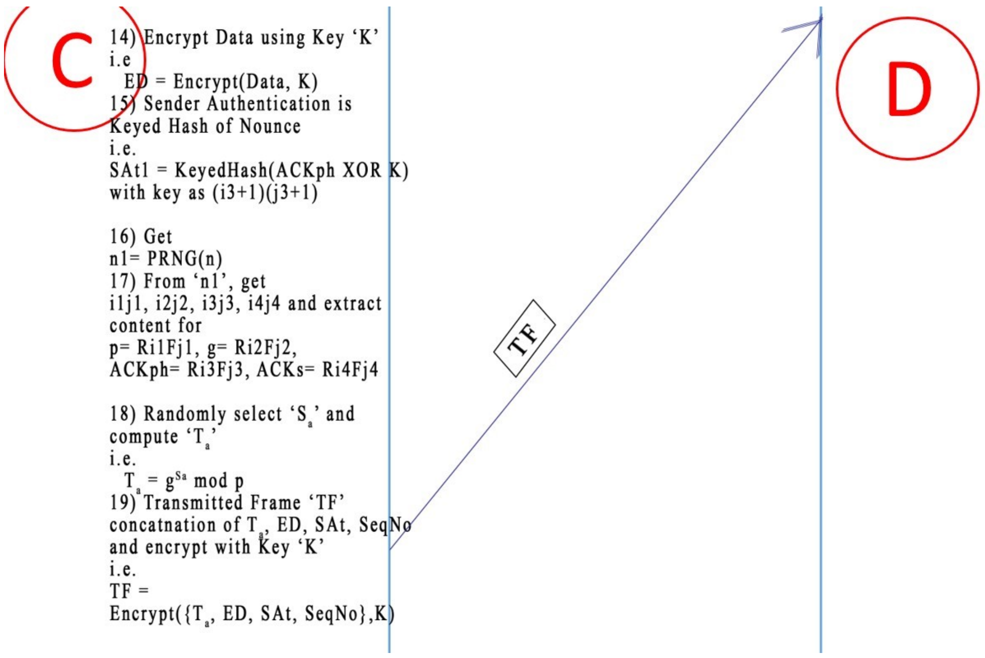

C
14) Encrypt Data using Key 'K' i.e ED = Encrypt(Data, K)
15) Sender Authentication is Keyed Hash of Nounce i.e. SAt1 = KeyedHash(ACKph XOR K) with key as (i3+1)(j3+1)
16) Get n1= PRNG(n)
17) From 'n1', get i1j1, i2j2, i3j3, i4j4 and extract content for p= Ri1Fj1, g= Ri2Fj2, ACKph= Ri3Fj3, ACKs= Ri4Fj4
18) Randomly select 'S_a' and compute 'T_a' i.e. $T_a = g^{Sa}$ mod p
19) Transmitted Frame 'TF' concatnation of T_a, ED, SAt, SeqNo and encrypt with Key 'K' i.e. TF = Encrypt({T_a, ED, SAt, SeqNo},K)
TF
D

Figure 20

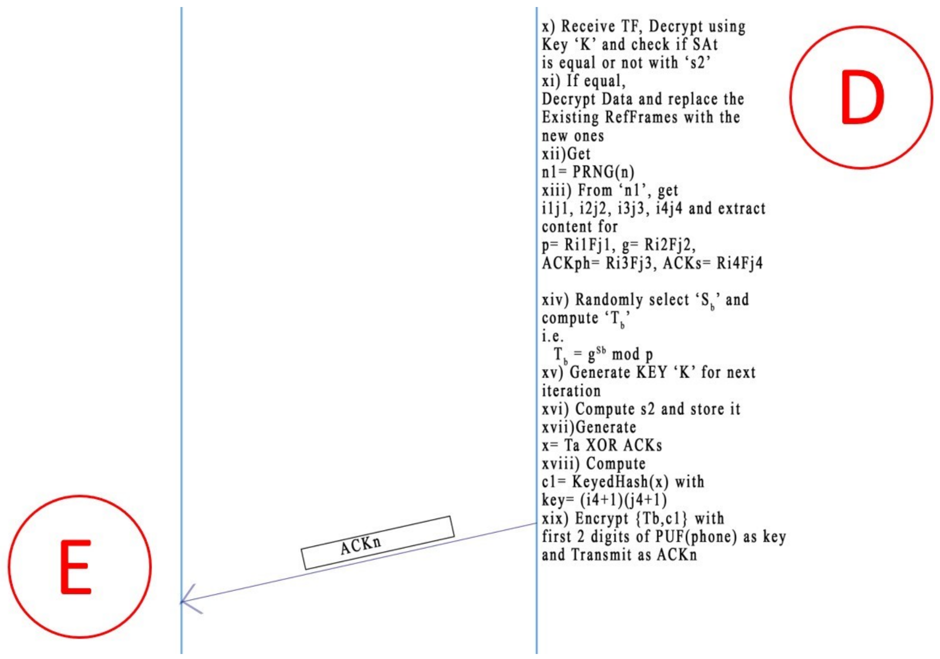

x) Receive TF, Decrypt using Key 'K' and check if SAt is equal or not with 's2'
xi) If equal, Decrypt Data and replace the Existing RefFrames with the new ones
xii)Get n1= PRNG(n)
xiii) From 'n1', get i1j1, i2j2, i3j3, i4j4 and extract content for p= Ri1Fj1, g= Ri2Fj2, ACKph= Ri3Fj3, ACKs= Ri4Fj4
xiv) Randomly select 'S_b' and compute 'T_b' i.e. $T_b = g^{Sb} \bmod p$
xv) Generate KEY 'K' for next iteration
xvi) Compute s2 and store it
xvii)Generate x= Ta XOR ACKs
xviii) Compute c1= KeyedHash(x) with key= (i4+1)(j4+1)
xix) Encrypt {Tb,c1} with first 2 digits of PUF(phone) as key and Transmit as ACKn
D
ACKn
E

Figure 21

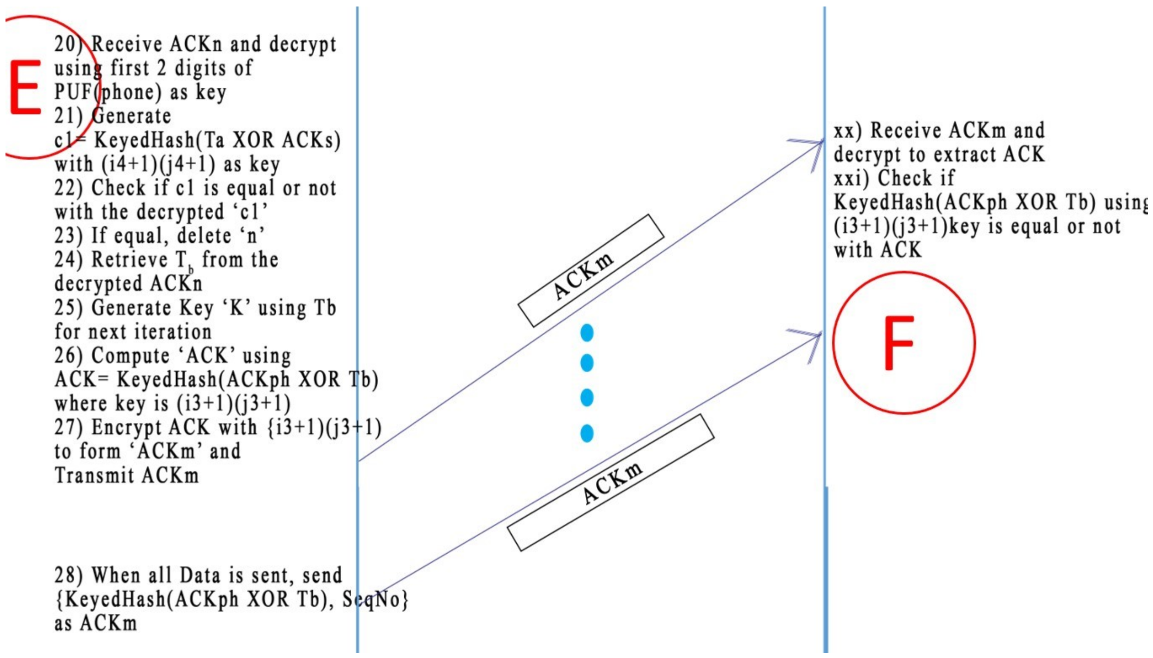

E
20) Receive ACKn and decrypt using first 2 digits of PUF(phone) as key
21) Generate c1= KeyedHash(Ta XOR ACKs) with (i4+1)(j4+1) as key
22) Check if c1 is equal or not with the decrypted 'c1'
23) If equal, delete 'n'
24) Retrieve T_b from the decrypted ACKn
25) Generate Key 'K' using Tb for next iteration
26) Compute 'ACK' using ACK= KeyedHash(ACKph XOR Tb) where key is (i3+1)(j3+1)
27) Encrypt ACK with {i3+1)(j3+1) to form 'ACKm' and Transmit ACKm
28) When all Data is sent, send {KeyedHash(ACKph XOR Tb), SeqNo} as ACKm
ACKm
ACKm
xx) Receive ACKm and decrypt to extract ACK
xxi) Check if KeyedHash(ACKph XOR Tb) using (i3+1)(j3+1)key is equal or not with ACK
F

Figure 22

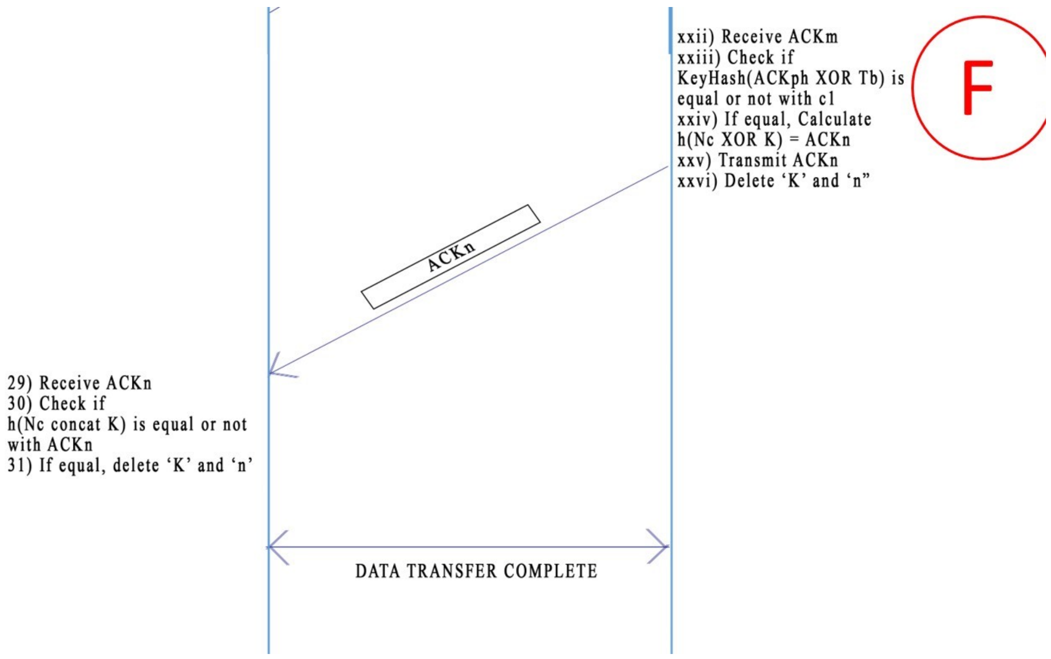


Figure 23

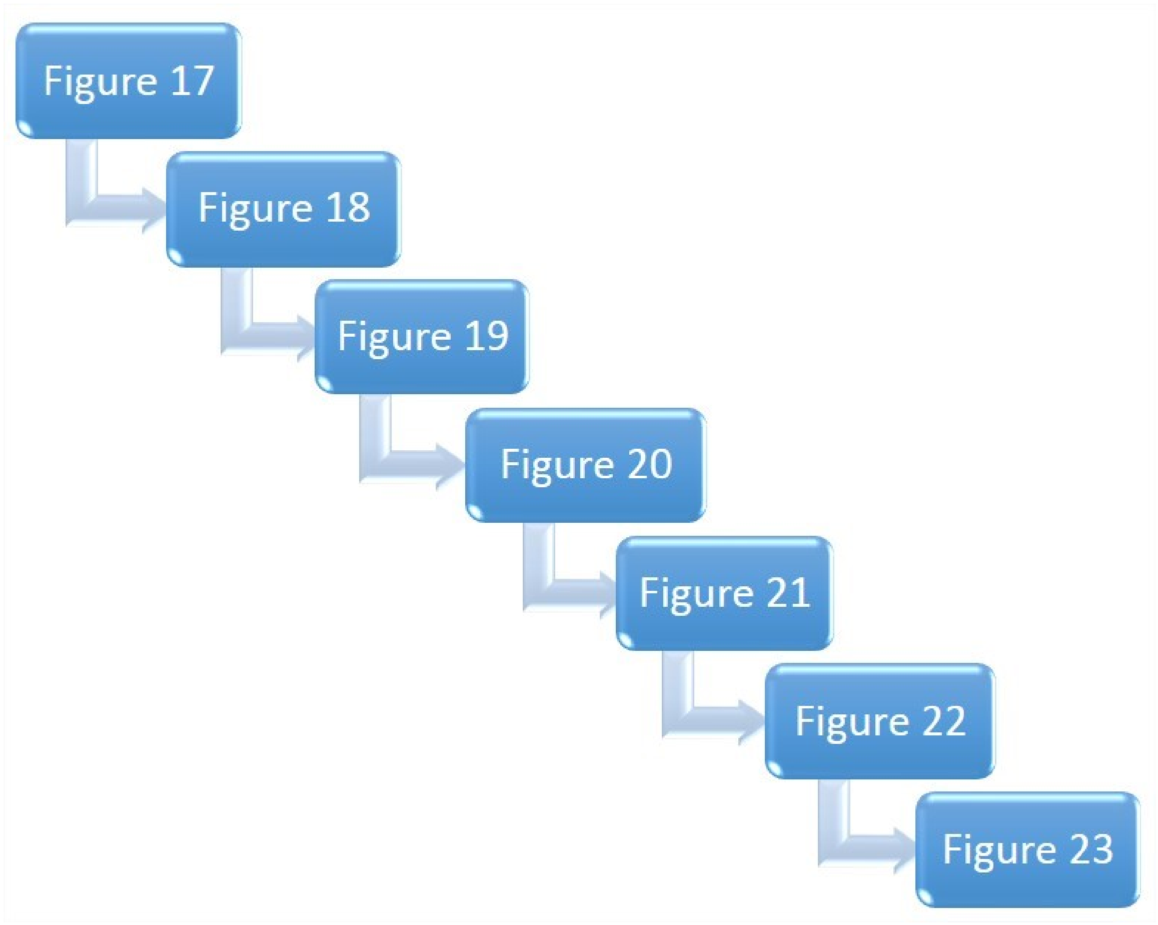


Figure 24 Data Transfer Phase

4.3 Application used for Data Collection to test the proposed approach

At the client end, the reference frames are formed/updated and then written into the file but at the server end, the reference frames are updated after the reception of the file. Figure 16 represents the layout and the contents of the Reference Frames. The fields in each of the frames are updated at both ends. But the fields possessing the sensor values are used as ‘p’, ‘g’ etc. which are used in the algorithm for key generation, authentication and acknowledgements etc.

The application used for the generation of the reference frames makes use of the sensors available in the smartphone. All the required methods and classes are implemented for retrieving the sensors’ values. These values are then written into a file in the text format with ‘.txt’ extension.

The researcher turned on the application developed and performed the test cases discussed in detail later in Chapter 6.

CHAPTER 5 IMPLEMENTATION

5.1 DEVELOPMENT ENVIRONMENT AND LIBRARIES USED

The entire proposed algorithm is implemented using Java platform on a Windows Operating System. The Android application was developed in Android Studio on the same Operating System. Table 2 shows the development environment used for implementing the proposed approach.

Programming Language	Java 1.8.0_25
Application Android minimum version	4.4 (Kitkat)
Operating System	Windows 8.1

Table 2 Development environment for the proposed approach

5.1.1 JAVA Environment

Java as a programming language gained popularity due its Write Once, Execute Anywhere feature. This made it platform independent and easily executable in any device irrespective of the Operating System upon which the device relies on. It is an object oriented programming i.e. everything is based on the classes defined in the program. The Java Virtual Machine (JVM) which gets installed in the system, is what actually makes Java as platform independent. Upon compilation of the java program, it yields a Bytecode whose execution is intended to be the JVM rather than the machine itself. Hence as long as the JVM is present in the system, the java snippet can be executed or compiled or both, on any platform.

Java is now an Oracle product but it was first released by James Gosling in 1995 under the roof of Sun Microsystems. It was in year 2009-2010, that Oracle acquired Sun microsystems. [45] The Table 3 shows the Java and IDE details which are used for this thesis work.

Java version	1.8.0_25
Netbeans IDE version	8.0.2

Table 3 Platform Specifications

There are many libraries in java which provide access to many facilities such as multi-threading, networking, security etc. The following are the packages which have been used in the implementation of the proposed approach;

- Security programming: The java.security package provides the classes, security services, name of the providers' (such as SHA-256, MD5 etc.) needed for facilitating the security aspect in java platform. This package contains classes that are not involved in securing the transmission content such as Message Digest and Signatures, which do not directly deal with encryption. 'getInstance()' function is used to get a specific provider. For instance, if we want to generate a Message Digest using the MD5 then its code snippet would be as follows:

  $MessageDigest\ md = MessageDigest.getInstance(\text{MD5});$ [47]

  The java.crypto package, contains classes that are involved in securing the transmission content such as Cipher and KeyAgreement, which directly deal with encryption. It has built-in providers for cryptographic algorithms such as DES, AES, ECDH etc. [47]

  Apart from the above, java.crypto.cert package provides Public Key Infrastructure PKI, which contains keys, certificates, public key encryptions, certificate authorities etc., needed for a secure exchange of data.[47]

- Network programming: The java.net package provides the classes and interfaces needed when we our program to transfer data from one machine to another machine across different platforms. In short, it helps multiple machines to communicate with each other. The process using which a TCP connection is established using the classes in this package are as follows:

  The ServerSocket class has methods which helps in accepting the incoming connection requests from various different sockets and allowing data to be transferred. The only argument which ServerSocket takes is the port number over which the communication shall take place.

Then a Socket is opened which shall be listening to all the incoming connection requests. It is this class that acts as a bridge for transferring the data from one end to the other. There is a Socket instance created at each end of the connection.

Once the Socket object at the one end, for instance; a server, is listening for the connection requests, another Socket at the other end, for instance; a client, tries to send the connection requests. The Socket at the client end creates an object by passing the IP address and the Port number on which the Server's socket is listening for the requests.

When the server's Socket instance receives the request, it accepts it and responds with a reference which indicates the commencement of the communication session.

During the communication, data is transferred and/or received using various classes such as DataOutputStream etc. & methods (from the Socket class) such as getInputStream() etc.

When the data transmission is completed, the close() method of the Socket class is used to end the session and the connection is terminated.[48]

- Collection Framework: The java.util package contains set of interfaces and classes which help in overcoming the limitation of the array structure i.e. its fixed size. Array sizes cannot be increased dynamically. Once declared, its size remains fixed for the entire span of the program. Collection/s, on the other hand, can grow dynamically. The object of the collection class stores the references of other objects while the collection interface has the abstract methods which represent the operations that can be performed on the objects. In short, this framework provides pre-packaged data structures and algorithm for its manipulations.

  Collection Interface has the List Interface, Map Interface, Set Interface etc. which can be manipulated independently.

  Collection Class has the ArrayList, LinkedList, HashMap, TreeSet etc. which are reusable data structure at its core.

The contents in them are inserted, deleted, altered etc. using the Iterator or Enumeration objects [49].

5.1.2 ANDROID Environment

The first version of Android was released in the market in the year 2007. Since then it has swept into almost every phone available in the market as it Operating System. Android was acquired by Google in the year 2005, two years post its foundation at Palo Alto in the year 2003. Its Open-source feature encouraged developer across the world, to embrace it and enrich it. Numerous mobile applications have been developed since its release. The current release is the Android 6.0 which is named as 'Marshmallow'. Basically, Android is Linux kernel based programming language for touchscreen devices [65].

Android Studio Version	1.4
Android SDK version	Android 4.0.3 (Icecream Sandwich) and above
Programming Language	Java 1.8.0_25
User interface	XML
Device	Samsung Galaxy S6
Device Android OS version	Android 5.0.1 (Lollypop)

Table 4 Application development environment specifications

The following is a brief overview on some of the modules available in the Android framework;

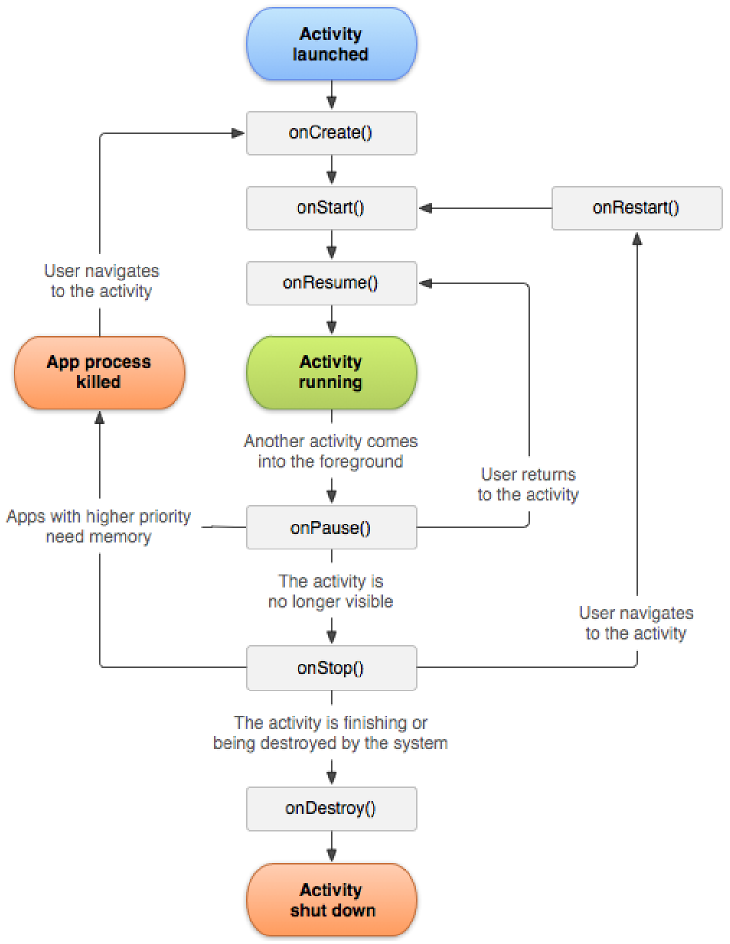
Activity Life cycle: Android uses a stack to handle and manipulate all of its foreground and background activities. The activity which is on the top of the stack is the activity which is running in the foreground, followed by the activity which was most recently sent to the background.

onCreate() and onDestroy() are the two ends of the activity lifecycle. The required resources ae allocation to the application in the onCreate() function and the very same resources are released when the onDestroy() method is called. All the functionalities necessary for the successful execution of the application runs inbetween these two function calls.

onStart() and onStop() are the two ends of the user interaction duration period. With the onStart() function call, the activity is made visible to the user to interact and with onStop() function call, the very same activity is hidden from the user. After the onStop() function call, the application may be in the background holding its resources but is moved from the top of the stack to the next position.

onResume() function call bring the activity back to the top of the stack and hence to the foreground for the user to interact again. onPause() function call sends it back to the background.

onCreate(), onDestroy() and onResume(), onPause() functions can be frequently called by an activity.[55]



Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Figure 25 Android activity life cycle [55]

Application framework: It handles all the API's which are imported by the developers. It maps the imported API's directly to the Hardware Abstraction Layer (HAL) interfaces. This framework, in short, is the starting point for application development. The services which are available in this framework are:

- ✓ *Sensor Manager* – This manager has the functionalities with which it controls the sensors available in the device. The Sensors include accelerometer, proximity, light etc. whose provision in the device varies from one manufacturer to another.
- ✓ *Wi-Fi Manager* – This manager has the functionalities which provides the information required for network connectivity. It provides details such as WI-FI signal strength, MAC address etc.
- ✓ *Activity Manager* – This manager is one of the most important as it controls the Application lifecycle and activity stack involved in the functioning of the developed application.
- ✓ *Content Providers* – This manager provides the functionalities which help in sharing the data and resources amongst application by linking them with one another.
- ✓ *Resource Manager* **-** This manager provides the functionalities required to handle the user color settings, interface layouts and strings displayed.
- ✓ *View System* – This mainly handles the functionalities concerned with the display/view of the application. It is primarily used for manipulation of the application's user interface aspects such as textview & editview sizes, buttons' placement or size or image etc.
- ✓ *Notifications Manager* – All the alerts and notifications are handled by this manager.

Hardware Abstraction Layer: This is automatically uploaded by the android system. This layer acts as an interface for functions implementation without accessing the higher layers.[55]

Linux Kernel: This is the bottommost layer in the architecture of android. Its primary responsibilities include system administration, process management, memory management and handling device drivers along with the equipment drivers such as drivers for camera, screen display etc.[55]

5.2 Implementation details of the phases in the proposed approach

The proposed approach constitutes of two phases; Registration Phase and Data Transfer Phase. The step by step discussion of each phase between the client application and the server, is clearly mentioned in Chapter 4. Here in this section, the implementation details of each phase is explained. The succeeding subsections discusses the major implemented operations upon which the entire proposed approach rotates around. Those major operations are:

- XOR operation
- Hash operation
- Keyed Hash operation
- Socket connection
- Random Number Generation
- ‘p’, ‘g’, ‘ACKph’, ‘ACKs’ etc. generation
- Diffie Hellman Keys generation
- RSA Key generation, Encryption & Decryption
- Reference Frame generation in Android application

5.2.1 XOR OPERATION.

XOR operation is performed numerous times in the proposed approach. For instance, in Registration Phase, the ‘z’ value is deduced with hashing of the outcome of the XOR operation between PUF of device and hashed fingerprint. Similarly, in Data Transfer phase, the seed for the random number generator is the outcome of the XOR operation between the ‘z’ value and the Nonce. The following snippet is used to perform the XOR operation between the byte array inputs and the outcome is stored in another byte array.

```
xor_result[pos] = (byte) (field1[pos] ^ field2[spos]);
```

where xor_result is a resulting byte array and field1 and field2 are the array values between whom the XOR operation is performed.

5.2.2 HASH OPERATION

Hash is a one-way operation i.e. if ‘A’ if hashed to get ‘B’ then ‘B’ cannot be decoded to get ‘A’. This makes it apt for authenticating the parties as only the legitimate owners’ would know the input of the hash. If the hashes do not match, then the sender is not authentic. For instance, as discussed in Section 4.2 of Chapter 4, the client application requests for fingerprint, generates a hash of it and deletes the original fingerprint. This helps in providing Access Control for the application in the proposed approach. Apart from this, hashing has been used multiple times in the proposed approach as it does not reveal which values are used as inputs.

For getting the Hash, MessageDigest class is used to perform a SHA-256 hashing. The outcome of the array of bytes is stored and is used as required by the proposed algorithm. The code snippet which performs the hash operation is as follows:

```
MessageDigest md = MessageDigest.getInstance("SHA-256");
md.update(hashed_image_byteArray);
byte[] hashed_image_bytes = md.digest();
```

5.2.3 KEYED HASH OPERATION

Hashing is secure, but to make it even more secure, Keyed Hash is used in the proposed approach. Keyed hash is mainly used for generation of the Acknowledgment packets in the proposed algorithm. SecretKeySpec class is used for which the provider ‘HmacSHA256’ is given as one of the arguments. The result is a byte array which is returned and stored. *javax.crypto.spec* library is imported to perform the mentioned operation. The following is the code snippet used for performing the Keyed Hash operation.

```
SecretKeySpec signingKey = new SecretKeySpec(hash_key.getBytes(),
"HmacSHA256");
Mac mac = Mac.getInstance("HmacSHA256");
mac.init(signingKey);
return mac.doFinal(message_to_be_hashed.getBytes());
```

5.2.4 SOCKET CONNECTION

Sockets are used for establishing a connection between the client and the server. Data is then transferred using the data writer and reader classes. The received data is then verified and the succeeding operations are performed accordingly. *java.net* package is used for implementing the entire socket connection.
At the server end, ServerSocket object is created on port 19 using the following code snippet.

```
ServerSocket server_Socket = new ServerSocket(19);
```

Then the following snippet is used to instantiate a Socket object which would listen to the incoming connection requests with the help of the ServerSocket class's 'accept()' function.

```
Socket channel_Socket = server_Socket.accept();
```

After the establishment of connection between the server and the device, in order to receive the incoming message, readLine() function of the BufferedReader class is used. The BufferedReader takes InputStreamReader object as its argument. The InputStreamReader in return takes the getInputStream() function of the Socket created, as its argument. The received message is stored into a String. The following snippet implements the above mentioned process.

```
BufferedReader receiver = new BufferedReader(new
InputStreamReader(channel_Socket.getInputStream()));
String incoming_msg = receiver.readLine();
```

The data which is to be sent out of the socket, is done by making use of the ObjectOutputStream class. It takes getOutputStream() function of the Socket class as its argument.

```
DataOutputStream sender = new
DataOutputStream(channel_Socket.getOutputStream());
sender.writeBytes("text to be sent");
```

When the objection of the socket session is completed or whenever there is an urge for the session to be terminated, the close() function of the Socket class is used.

```
smartphone_Socket.close();
```

5.2.5 RANDOM NUMBER GENERATION

Usually the Random class of the *java.util* library can be used for Random Number generation. But when we want to generate a Random Number in a secure fashion, then we make use of the subclass of the Random class. SecureRandom is a sub-class of the Random class which generates a random number which is strong cryptographically as specified in RFC-1750: Randomness Recommendations for Security. The random number thus generated satisfies the tests specified in FIPS 140-2, Security Requirements for Cryptographic Modules[50]. The following code snippet is used to generate a random number required for the proposed approach.

```
SecureRandom prng = new SecureRandom(new_seed);
int n = prng.nextInt();
```

5.2.6 'p', 'g', 'ACKph', 'ACKs' TERMS GENERATION

The value of 'n' which is generated using the pseudo random number generator is the crux for the generation of the 'p', 'g' which are used for the Key Generation in the Diffie Hellman algorithm and for the generation of 'ACKph', 'ACKs' which are used as acknowledgements in-between the client and the server.
The generated random number is split and stored into an ArrayList object named 'pointer'. This ArrayList object's contents are then used to retrieve the values stored in the Reference

Frames arrays. *java.util* package is imported to avail the facilities of the ArrayList, Iterator classes etc. The following snippet shows how the random number ‘n’ is split and stored into the pointer ArrayList.

```
String n_string = String.valueOf(n); //random number ‘n’ is converted
                                          to String for splitting
String[] n_arr = n_string.split("");//split and stored into an array
                                          format
int looper=0;
do {
pointer.add(Integer.parseInt(n_arr[looper]));//each digit of ‘n’ is
                                    stored into ArrayList ‘pointer’
looper++;
} while (pointer.size() < n_arr.length);
```

Once the digits of the random number are stored into the ArrayList, they are then accessed and the respective locations’ content from the Reference Frames are retrieved. For instance, as discussed in Section 4.2, the first two digits of the random number generated are used as pointers to the location whose contents is used as the value for ‘p’. The following code snippet illustrates how the mentioned example can be achieved;

p = (get_Reference_Frame_Content(pointer.get(0), pointer.get(1)));
//where get_Reference_Frame_Content is a method which takes the array indices as parameters. The first parameter refers to the Reference Frame number and the second frame represents the Field number in the chosen Reference Frame.

```
//get_Reference_Frame_Content() pseudo code
get_Reference_Frame_Content(int i, int j) {
double content;
if (i > (row_count - 1)) {//loops it around the reference frames used
i = i % (row_count - 1);
}
if (j > 4) {//loops it around the fields used
j = j % 4;
}
content = reference_frames[i][j];
return content;
}
```

Similarly, the values of ‘g’, ‘ACKph’, ‘ACKs’ are deduced and stored in their respective variable and utilized as needed by the proposed approach.

```
g = (get_Reference_Frame_Content(pointer.get(2), pointer.get(3)));
ACKph = (get_Reference_Frame_Content(pointer.get(4), pointer.get(5)));
ACKs = (get_Reference_Frame_Content(pointer.get(6), pointer.get(7)));
```

5.2.7 DIFFIE HELLMAN KEY GENERATION

Diffie Hellman algorithm requires ‘p’, ‘g’, ‘S_a’, ‘S_b’ and ‘T_a’, ‘T_b’ for the generation of its key. The value of ‘S_a’ & ‘S_b’ are Random numbers, ‘T_a’ & ‘T_b’ are calculated using the formula (i) & (ii) and the SecretKey is generated using the formula (iii) & (iv) asgiven below;

i. $Ta = g^{Sa} \, mod \, p$
ii. $Tb = g^{Sb} \, mod \, p$
iii. Secret $Key = Tb^{Sa} \, mod \, p$
iv. Secret $Key = Ta^{Sb} \, mod \, p$

For attaining the above mentioned, the ‘T_a’, ‘T_b’ and the Secret Key are initialized as BigIntegers which is obtained from the *java.math* package. The following code snippet depicts how the value of ‘T_a’ & ‘T_b’ are calculated;

```
T = (g).pow(S.intValue());
T = T.mod(p);
```

Similarly, the following code snippet helps in calculation of the Secret Key;

```
key = (T).pow(S.intValue());
key = key.mod(p);
```

5.2.8 RSA KEY GENERATION, ENCRYPTION & DECRYPTION

The message which has to be transmitted, is composed and hashed as discussed in Section 4.2. This hashed version of the message is the Message Digest. This when encrypted acts as the Digital Signature. The whole message along with the digital signature is encrypted using the public key of the server. The entire message is secured using the RSA encryption.

Following is the implementation details of the RSA algorithm.
Two important libraries are used, mainly for handling the BigInteger and for generating a secure random number. They are *java.math* and *java.security.SecureRandom* respectively.

As discussed in Section 2.1, the keys required for the RSA algorithm are generated after following a sequence of calculations. For the proposed approach, the keys are generated using the following snippet:

```
//calculating p*q
n = p.multiply(q);
//calculating  (p-1)*(q-1)
BigInteger m =
 (p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));
e = new BigInteger("3");
while (m.gcd(e).intValue() > 1) {
e = e.add(new BigInteger("2"));
}
d = e.modInverse(m);
//public key ( e, n ) and private key ( d, n )
```

For encryption, the data to be sent is processed as discussed in Section 2.1. In the proposed approach, data is encrypted using the following snippet;

```
public synchronized String encrypt(String plaintext) {
return (new BigInteger(plaintext.getBytes())).modPow(e, n).toString();
}
```

For decryption, the received data is processed as discussed in Section 2.1. In the proposed approach, data is decrypted using the following snippet

```
public synchronized String decrypt(String ciphertext) {
return new String((new
BigInteger(ciphertext)).modPow(d,n).toByteArray());
}
```

5.2.9 REFERENCE FRAME GENERATION IN ANDROID APPLICATION

As discussed in Section 4.2, the Reference Frames are stored in an array format. The Rows represent the number of Reference Frames used and the columns store the Sensor values which are put into the desired use. The Reference Frames' contents are accumulated and stored in the required format via the mobile application. For gathering the sensor values, following are the set of code snippet that were utilized:
The SensorManager class in the Android system is used to activate and/or manipulate the sensors available in the smartphone/device. In the following snippet, SensorManager is the class, getSystemService() is the function call whose parameter is the sensor service of the current Context.

```
SensorManager sensor_manager =
(SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

Then the sensor is instantiated using the getDefaultSensor() of the sensor manager class.

Following is the snippet for Pressure sensor.

```
Sensor pressure_sensor =
sensor_manager.getDefaultSensor(Sensor.TYPE_PRESSURE);
```

Then the obtained sensor is activated using the registerListener() function call. This function takes the context, sensor and the delay as parameters. Following is the snippet which registers the Pressure sensor.

```
sensor_manager.registerListener(MainActivity.this,
pressure_sensor, SensorManager.SENSOR_DELAY_NORMAL);
```

Similarly, to stop a sensor from listening to changes within its hardware, the following snippet is used where the context and the sensor is used as parameters to unregisterListener() function:

```
sensor_manager.unregisterListener(MainActivity.this, light_sensor);
```

After registering the sensor, the onSensorChanged() function which takes the SensorEvent's object as it parameter, is over ridden. This method identifies any change that occurs in the sensor's hardware and that change is what represents the sensor value. The following code snippet displays the function for Pressure sensor;

```
@Override
public void onSensorChanged(SensorEvent event) {
if (event.sensor.getType() == Sensor.TYPE_PRESSURE)
{
pressure_values = event.values[0];
//code to write to file
}}
```

The recorded values are then written into a text file using the BufferedWriter and FileWriter objects.

```
BufferedWriter writer = new BufferedWriter((new
FileWriter("RFFile.txt", true));
writer.write(pressure_values+ "&");//& is used as an identifier while
retrieving the values at server end
writer.flush(); writer.close();
```

For the generation of the Reference Values at the client end, the following code snippet is used from within a nested for loop. This For loop is basically used to handle the Array Structure.

```
reference_frames[i][j] = pressure_values;
```

At the Server end, the values are read from the file which is sent from the client and is split using the '&' as an identifier. The split values are then stored into a two dimensional array. Following code snippet gives a brief overview of the implementation.

```
space_split_array = sub.split("&");
//for loop
reference_frames[i][j] = Double.parseDouble(space_split_array[k]);
```

5.2.10 PERMISSIONS REQUIRED FOR ANDROID APPLICATION

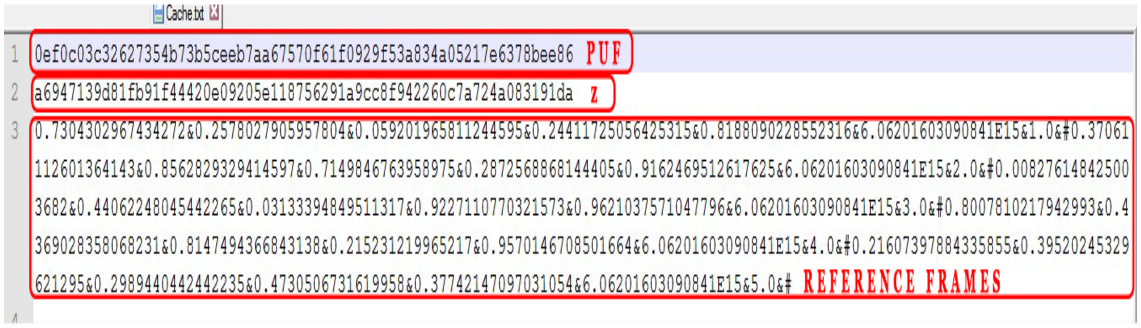
Permissions are needed in order for the application to function properly, collect the data appropriately and send them to the remote server. The following are the permissions which are used in the application developed.

```
<uses-permission android:name=
      "android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name = "android.permission.BODY_SENSORS" />
<uses-permission android:name = "android.permission.READ_PHONE_STATE"
/>
<uses-permission android:name=
      "android.permission.WRITE_INTERNAL_STORAGE"/>
<uses-permission android:name=
      "android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name=
      "android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
/>
<uses-permission android:name=
      "android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
/>
```

CHAPTER 6 EXPERIMENTAL RESULTS AND ANALYSIS

The Sections in this chapter discusses about the Experimental Setup, the Test scenarios with their results and finally we compare it with the existing algorithm. The main objective of the proposed approach is to make the keys generated by the parties as un-predictable as possible to minimize the probability of occurrence of the MITM attack in the Diffie Hellman algorithm. Hence different test cases are designed which mimic the real time scenarios under which the Reference Frames which are used for key generation, is collected. The keys thus generated are analyzed to check if there is any noticeable pattern among them. The whole approach is also tested using different Reference Frame count for different file counts as well.

Figure 26 shows the content that is stored at each of the client and the server before the commencement of the Data Transfer phase, post Registration Phase.



```
0ef0c03c32627354b73b5ceeb7aa67570f61f0929f53a834a05217e6378bee86
a6947139d81fb91f44420e09205e118756291a9cc8f942260c7a724a083191da
0.7304302967434272&0.2578027905957804&0.05920196581124459 5&0.24411725056425315&0.8188090228552316&6.06201603090841E15&1.0&#0.37061
112601364143&0.8562829329414597&0.7149846763958975&0.2872568868144405&0.9162469512617625&6.06201603090841E15&2.0&#0.00827614842500
3682&0.44062248045442265&0.03133394849511317&0.9227110770321573&0.9621037571047796&6.06201603090841E15&3.0&#0.8007810217942993&0.4
369028358068231&0.8147494366843138&0.215231219965217&0.9570146708501664&6.06201603090841E15&4.0&#0.21607397884335855&0.39520245329
621295&0.2989440442442235&0.4730506731619958&0.37742147097031054&6.06201603090841E15&5.0&#
```

Figure 26 Entities stored at each of the parties involved in the communication

The data which is sent over the wire in between the client and the server programs are captured using Wireshark tool and the captured traffic is also analyzed. A protocol analysis tool named Scyther is used to evaluate the protocol in the proposed approach. This protocol analyzer tool displays all of the possible attacks in the proposed approach, if any. This was used to ensure that this work was not susceptible to any major known attacks.

This chapter is organized as follows, we begin with the Experimental Setup discussing about the devices with the environment under which the data is collected and the tools used for evaluation. This section is followed by a set of sections which elaborates on various test cases considered and experimented upon. Section 6.2 deals with the evaluations done using Wireshark followed by Section 6.3 which deals with the various real time test cases. Section 6.4 shows the evaluations performed using Scyther and Sections 6.5 & 6.6 which present the performance & security analysis respectively. Finally, we conclude with the summary discussing the outcome of the experiments performed and the results thus obtained.

6.1 EXPERIMENTAL SETUP

The data required for testing and evaluating the proposed approach is accumulated via a mobile application which was installed in a smartphone mobile device. The device specification is as follows[51]:

- Device Model: Samsung Galaxy S6
- Android Version: Android 5.0.2 (Lollypop)
- RAM: 3GB
- CPU: Quad-core 1.5 GHz Cortex-A53 & Quad-core 2.1 GHz Cortex-A57
- Chipset: Exynos 7420 Octa
- Internal Memory: 32GB

The client and the server instances are implemented and executed using Java programming language in a personal laptop. The specification of the laptop is as follows:

- Operating System: Windows OS 8.1
- RAM: 8GB
- HardDisk Capacity: 500GB
- Java Version: Java 1.8.0_25
- Processor: Intel® Core™ i5-4200U CPU @ 1.60GHz 2.29GHz
- System Type: 64-bit Operating System, x64-based processor.

The mobile application is installed in the device and the researcher performed the activities required for each of the test cases. The data thus generated is then transferred to the laptop and the Client-Server socket program is executed.

All the keys generated are stored into a text file for analysis. The time taken for the execution to complete is noted down and plotted into a graph to understand the time consumption pattern. Also the generated keys are then plotted into a map to observe for any possible patterns.

During the execution of the java programs, the encryption of the data packet, which is to be transmitted, is avoided to make the contents visible for analysis. Only the Data part is encrypted, rest of the packet is left un-encrypted and is sent as plain text. Added to that, since sending such huge files using sockets is not feasible, we have encrypted only the file name and transferred the encrypted part as Data. At the same time, we have stored a copy of the encrypted file which is decrypted later by the other party after the obtaining the file name with the use of proper decryption key. RawCap is used to capture the loopback traffic generated by the data exchanged between the socket programs[52]. This captured traffic is then analysed using a traffic analyzer tool, 'Wireshark'[53].

- Wireshark Version: 1.12.11
- RawCap Version: 0.1.5.0

The entire proposed protocol is then evaluated using a protocol testing tool 'Scyther'. This tool is run in Linux based system. The Scyther specification is as follows;

- Scyther Version: 1.1.3 [54][62]

Only the following sensors are chosen for Reference Frame for the study purpose, as these are known to have relatively less probability of zero value generation. For instance, the proximity sensor value is either numeric Zero or numeric Eight, where numeric Eight represents the presence of the device away from the ear and numeric Zero, represents the presence of device near the ear. Thus the list of sensors considered for the testing are as follows:

- Pressure Sensor
- Gravity Sensor
- GPS Sensor

- Orientation Sensor
- Gyroscope Sensor

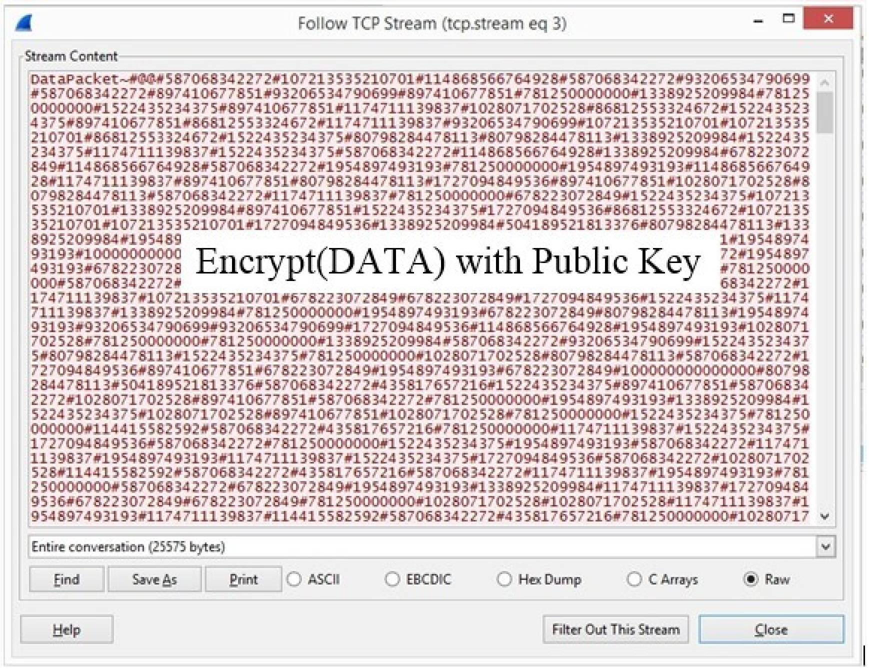
Each of the test cases are triggered for different count of Reference Frames, to analyze their execution times. The following are the number of Reference Frames considered for each of the test cases;

- Five Reference Frames
- Ten Reference Frames
- Fifteen Reference Frames
- Twenty Reference Frames
- Twenty Five Reference Frames

6.2 Evaluation using Wireshark

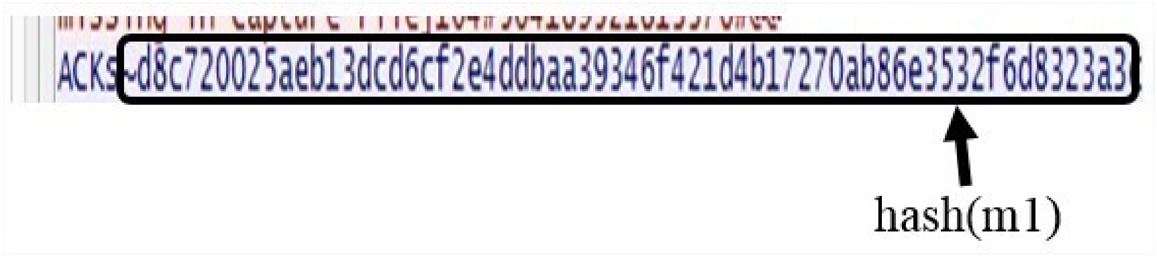
The actual proposed approach transmits most of the message in an encrypted form. This encrypted message is then decrypted at the corresponding other end using the appropriate key. To understand what is sent into the communication channel, this message is not encrypted. Only the actual data is encrypted and sent into the communication channel.

The packets are captured using ‘RawCap’ as discussed in Section 6.1 and analyzed using ‘Wireshark’. The following is the snapshot of the traffic captured during Registration Phase.


Follow TCP Stream (tcp.stream eq 3)
Stream Content
Encrypt(DATA) with Public Key
Entire conversation (25575 bytes)
Find
Save As
Print
ASCII
EBCDIC
Hex Dump
C Arrays
Raw
Help
Filter Out This Stream
Close

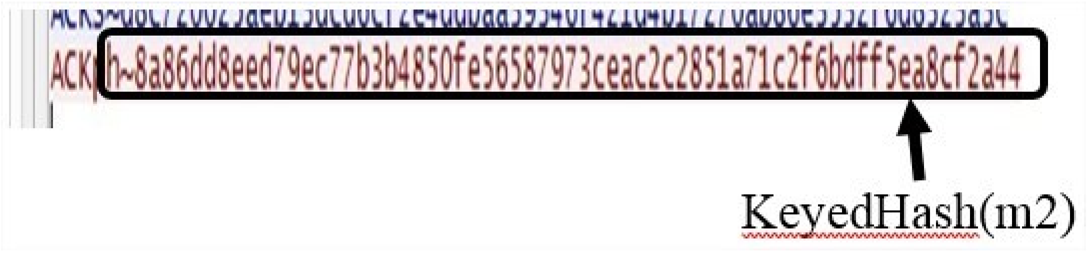
```
DataPacket~#@@#587068342272#1072135352107 01#114868566764928#587068342272#93206534790699
#587068342272#897410677851#93206534790699#897410677851#781250000000#1338925209984#78125
0000000#1522435234375#897410677851#1174711139837#1028071702528#868125533246 72#152243523
4375#897410677851#868125533246 72#1174711139837#93206534790699#1072135352107 01#107213535
210701#868125533246 72#1522435234375#8079828447 8113#8079828447 8113#1338925209984#1522435
234375#1174711139837#1522435234375#587068342272#114868566764928#1338925209984#678223072
849#114868566764928#587068342272#1954897493193#781250000000#1954897493193#1148685667649
28#1174711139837#897410677851#8079828447 8113#1727094849536#897410677851#1028071702528#8
0798284478113#587068342272#1174711139837#781250000000#678223072849#1522435234375#107213
535210701#1338925209984#897410677851#1522435234375#1727094849536#868125533246 72#1072135
35210701#107213535210701#1727094849536#1338925209984#504189521813376#80798284478113#133
8925209984#195489                                                            1#19548974
93193#1000000000                                                            72#1954897
493193#6782230728                                                           #781250000
000#587068342272#                                                           68342272#1
174711139837#107213535210701#678223072849#678223072849#1727094849536#1522435234375#1174
711139837#1338925209984#781250000000#1954897493193#678223072849#80798284478113#19548974
93193#93206534790699#93206534790699#1727094849536#114868566764928#1954897493193#1028071
702528#781250000000#781250000000#1338925209984#587068342272#93206534790699#152243523437
5#80798284478113#1522435234375#781250000000#1028071702528#80798284478113#587068342272#1
727094849536#897410677851#678223072849#1954897493193#678223072849#100000000000000#80798
284478113#504189521813376#587068342272#435817657216#1522435234375#897410677851#58706834
2272#1028071702528#897410677851#587068342272#781250000000#1954897493193#1338925209984#1
522435234375#1028071702528#897410677851#1028071702528#781250000000#1522435234375#781250
000000#114415582592#587068342272#435817657216#781250000000#1174711139837#1522435234375#
1727094849536#587068342272#781250000000#1522435234375#1954897493193#587068342272#117471
1139837#1954897493193#1174711139837#1522435234375#1727094849536#587068342272#1028071702
528#114415582592#587068342272#435817657216#587068342272#1174711139837#1954897493193#781
250000000#587068342272#678223072849#1954897493193#1338925209984#1174711139837#172709484
9536#678223072849#678223072849#781250000000#1028071702528#1028071702528#1174711139837#1
954897493193#1174711139837#114415582592#587068342272#435817657216#781250000000#10280717
```

Figure 27 Encrypted Data during the Registration phase.



```
ACKs~d8c720025aeb13dcd6cf2e4ddbaa39346f421d4b17270ab86e3532f6d8323a3
```

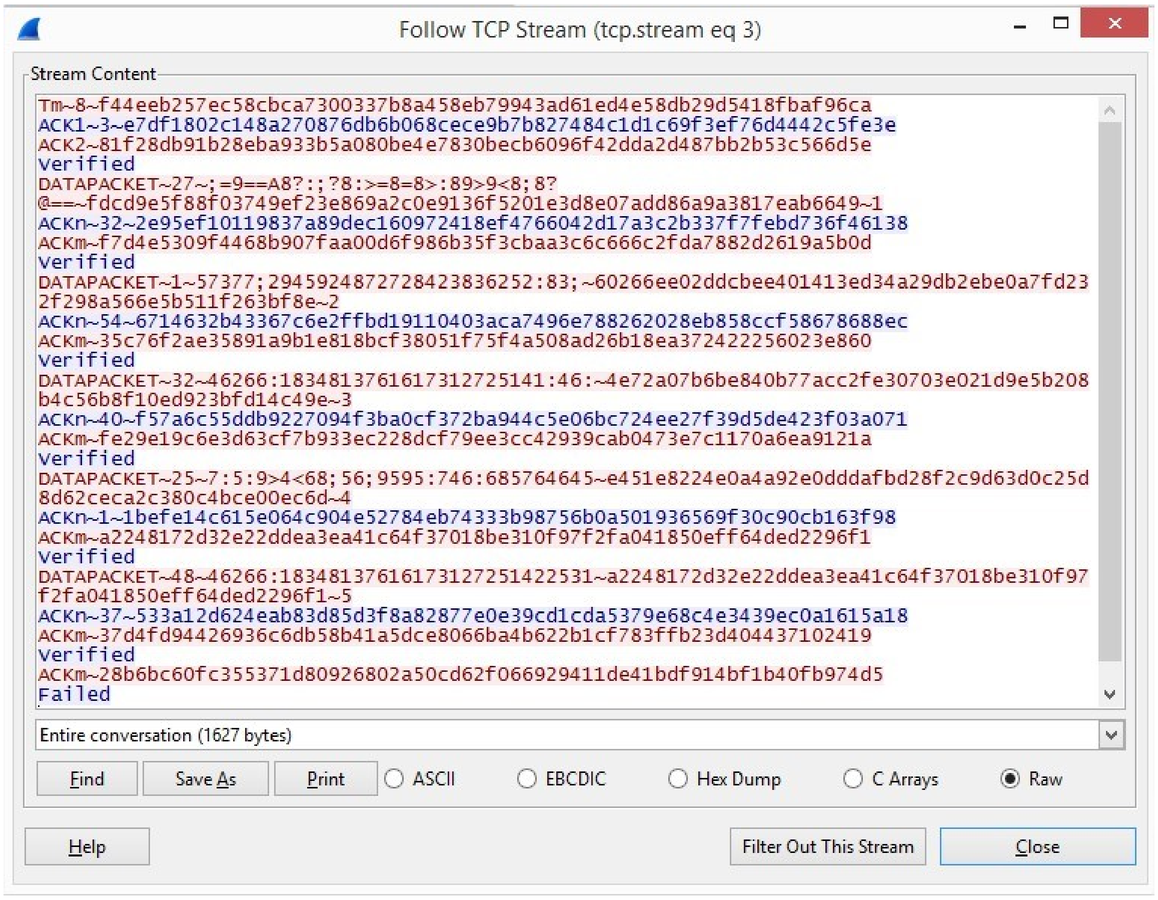
Figure 28 Second Message during the Registration phase.



```
ACKh~8a86dd8eed79ec77b3b4850fe56587973ceac2c2851a71c2f6bdff5ea8cf2a44
```

Figure 29 Third Message during the Registration phase.

The following is the snapshot of the traffic captured during the Data Transfer Phase.



```
Tm~8~f44eeb257ec58cbca7300337b8a458eb79943ad61ed4e58db29d5418fbaf96ca
ACK1~3~e7df1802c148a270876db6b068cece9b7b827484c1d1c69f3ef76d4442c5fe3e
ACK2~81f28db91b28eba933b5a080be4e7830becb6096f42dda2d487bb2b53c566d5e
Verified
DATAPACKET~27~;=9==A8?:;?8:>=8=8>:89>9<8;8?
@==~fdcd9e5f88f03749ef23e869a2c0e9136f5201e3d8e07add86a9a3817eab6649~1
ACKn~32~2e95ef10119837a89dec160972418ef4766042d17a3c2b337f7febd736f46138
ACKm~f7d4e5309f4468b907faa00d6f986b35f3cbaa3c6c666c2fda7882d2619a5b0d
Verified
DATAPACKET~1~57377;294592487272842383625 2:83;~60266ee02ddcbee401413ed34a29db2ebe0a7fd23
2f298a566e5b511f263bf8e~2
ACKn~54~6714632b43367c6e2ffbd19110403aca7496e788262028eb858ccf58678688ec
ACKm~35c76f2ae35891a9b1e818bcf38051f75f4a508ad26b18ea37242225 6023e860
Verified
DATAPACKET~32~46266:1834813761617312725141:46:~4e72a07b6be840b77acc2fe30703e021d9e5b208
b4c56b8f10ed923bfd14c49e~3
ACKn~40~f57a6c55ddb9227094f3ba0cf372ba944c5e06bc724ee27f39d5de423f03a071
ACKm~fe29e19c6e3d63cf7b933ec228dcf79ee3cc42939cab0473e7c1170a6ea9121a
Verified
DATAPACKET~25~7:5:9>4<68;56;9595:746:685764645~e451e8224e0a4a92e0dddafbd28f2c9d63d0c25d
8d62ceca2c380c4bce00ec6d~4
ACKn~1~1befe14c615e064c904e52784eb74333b98756b0a501936569f30c90cb163f98
ACKm~a2248172d32e22ddea3ea41c64f37018be310f97f2fa041850eff64ded2296f1
Verified
DATAPACKET~48~46266:18348137616173127251422531~a2248172d32e22ddea3ea41c64f37018be310f97
f2fa041850eff64ded2296f1~5
ACKn~37~533a12d624eab83d85d3f8a82877e0e39cd1cda5379e68c4e3439ec0a1615a18
ACKm~37d4fd94426936c6db58b41a5dce8066ba4b622b1cf783ffb23d404437102419
Verified
ACKm~28b6bc60fc355371d80926802a50cd62f066929411de41bdf914bf1b40fb974d5
Failed
```

Entire conversation (1627 bytes)

( ) ASCII ( ) EBCDIC ( ) Hex Dump ( ) C Arrays [✓ Raw]

Figure 30 Traffic content during the Data Transfer phase.

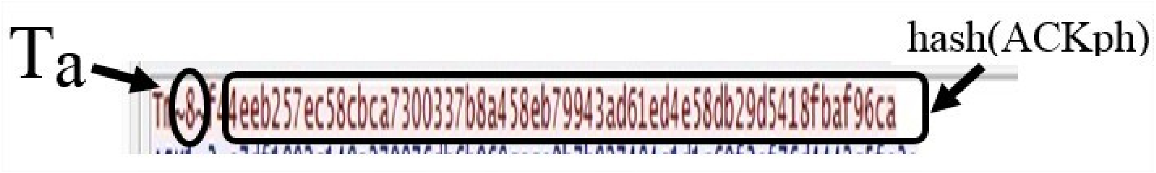

Ta
hash(ACKph)
Tm~8~f44eeb257ec58cbca7300337b8a458eb79943ad61ed4e58db29d5418fbaf96ca

Figure 31 First packet transferred in the Data Transfer Phase

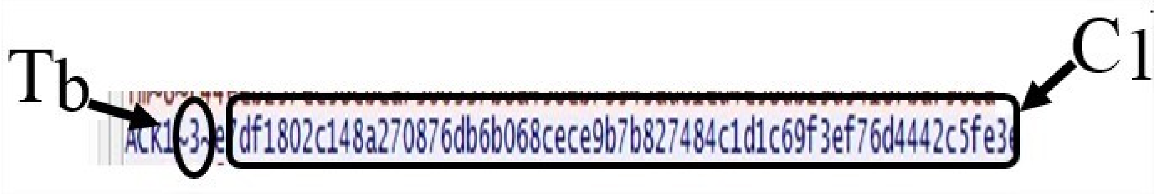
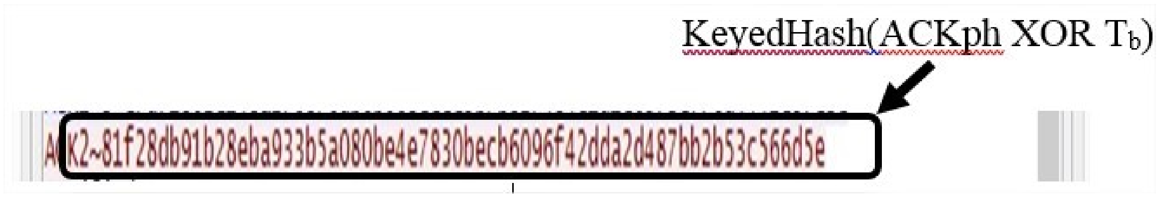
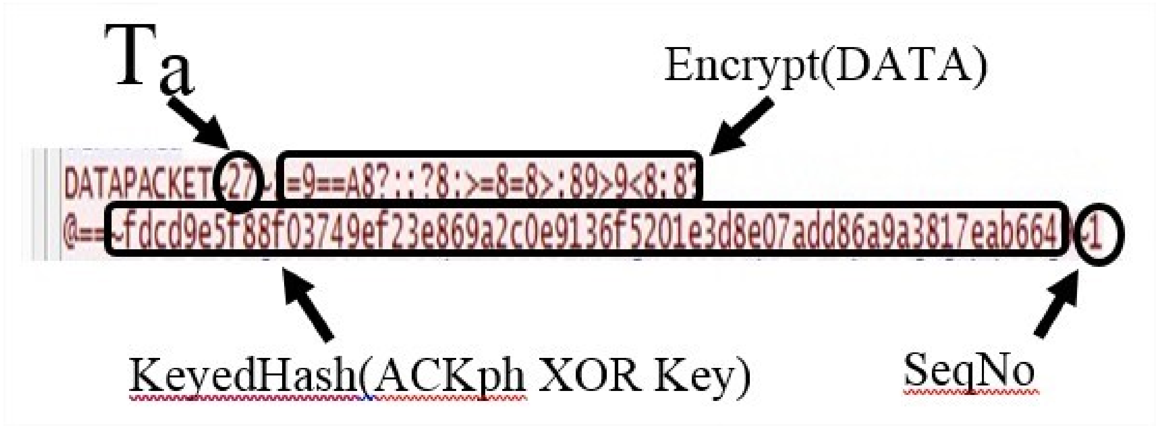

Tb
C1
ACK1~3~e7df1802c148a270876db6b068cece9b7b827484c1d1c69f3ef76d4442c5fe3

Figure 32 Second packet transferred in the Data Transfer Phase



```
ACK2~81f28db91b28eba933b5a080be4e7830becb6096f42dda2d487bb2b53c566d5e
```

Figure 33 Third packet transferred in the Data Transfer Phase



```
DATAPACKET~27~=9==A8?::?8:>=8=8>:89>9<8:8?
@==~fdcd9e5f88f03749ef23e869a2c0e9136f5201e3d8e07add86a9a3817eab664~1
```

Figure 34 Fourth packet transferred in the Data Transfer Phase

Among all the contents which constituted the messages that were exchanged, only ‘Ta’ and ‘Tb’ are the factors which are a part of the key generation process. But these values by themselves cannot contribute to the generation of the Secret Key without the support of ‘p’, ‘g’ and their respective ‘Sa’ and ‘Sb’ values. Added to that, the actual data “351559072370265_1406201600062346” is encrypted and the outcome is shown in the Figure 34. It is highly unlikely to revert back to the actual data without the knowledge of the actual Secret Key that is used to encrypt the data. Thus it is clear that the prediction of keys is relatively difficult even if the hacker intercepts the ongoing communication.

6.3 Evaluation using test scenarios

The test scenarios considered for this research work depict the day-to-day activities of the users. We placed the smartphone device in different positions which would imitate the position of the device if it were to be in a real time scenario. The testing began with Simulated data collection i.e. the contents in the Reference Frames and the data files were generated with the help of a Pseudo Random number generator. Then the developed mobile application was put into action under various test cases. The following is the list of all the test cases under which the mobile application accumulated the data required for testing the proposed approach.

- When the device is moved ‘Vertically’
- When the device is in an ‘Idle’ position
- When the User is ‘Walking’ with the device in his trousers’ pocket
- When the User is inside an ‘Elevator’ with the device in his trousers’ pocket
- When the User is walking up/down the ‘Stairs’ with the device in his trousers’ pocket.
- When the User is within a ‘Vehicle’ with the device in his trousers’ pocket

6.3.1 USING THE ‘SIMULATED DATA’

In this test case, the proposed approach is tested using the data generated by a random number generator in a Java program. The upper - lower bounds for each sensor is set and the values are generated. Even the sensors are also selected at random. In this test case, no real sensor is used. Only a reference of the real sensor is duplicated and tused. The purpose of this test case is to check if this approach can be utilized in devices which do not include any Sensors’ hardware.

The generated values and the files are then used as input for the proposed approach. The communication is initiated and the Keys generated are written into a file. The keys in the file are then plotted into a graph and analyzed.

The Keys generated and the Execution times are recorded for the Reference Frames count mentioned in Section 6.1 for different file counts i.e. in the first execution, five DataFiles were transferred between the client and the server socket using each of the Reference Frames count scenario. Then in the second run, ten DataFiles were transferred again between the sockets using each of the Reference Frames count scenario and so on until the DataFile count was observed to show an exponential spike in the execution time or until it shows that the execution times are remaining constant irrespective of the Reference frames count used. The count of data files is five, ten, fifteen, twenty, twenty-five and fifty.

The following is the image of the Graph which represents the execution time taken for different DataFile count with different Reference Frames.

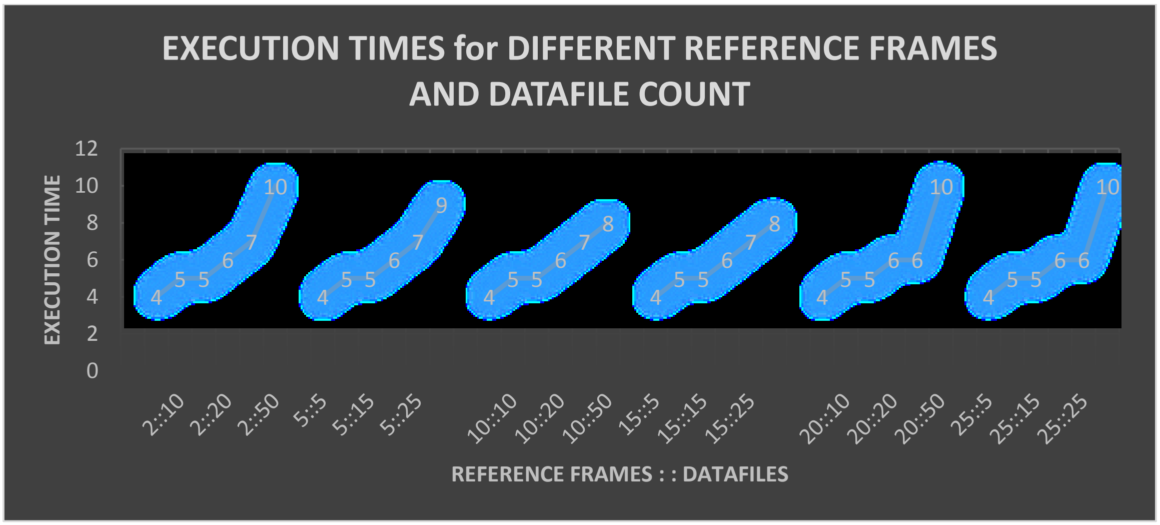


Figure 35 Execution Time vs ReferenceFrames count & DataFile count for Simulated Data

Figure 36 shows the graphs with the key values that are obtained for different DataFile count but with five Reference Frames. Each graph (in Figures 84-89) representing the keys generated for different Data File counts are displayed individually in Appendix A.

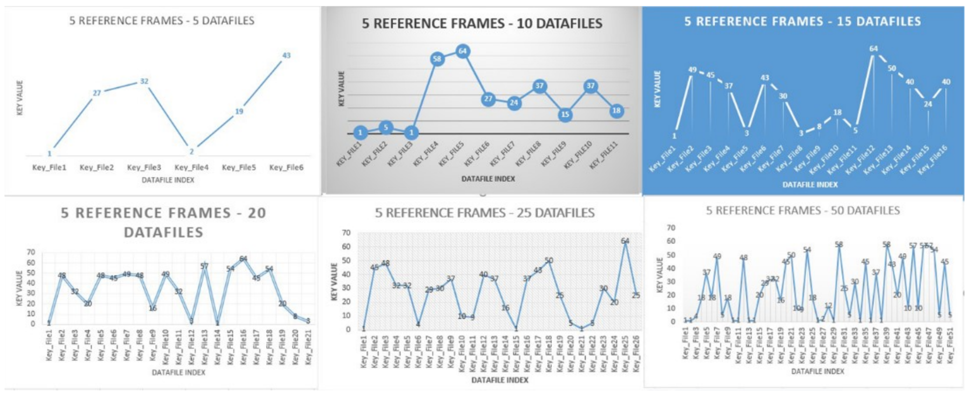


Figure 36 Key values obtained for each Data file with 5 Reference Frames for Simulated Data

Figure 37 shows the graphs with the key values that are obtained for different DataFile count but with ten Reference Frames. Each graph (in Figure 90-95) representing the keys generated for different Data File counts are displayed individually in Appendix A.

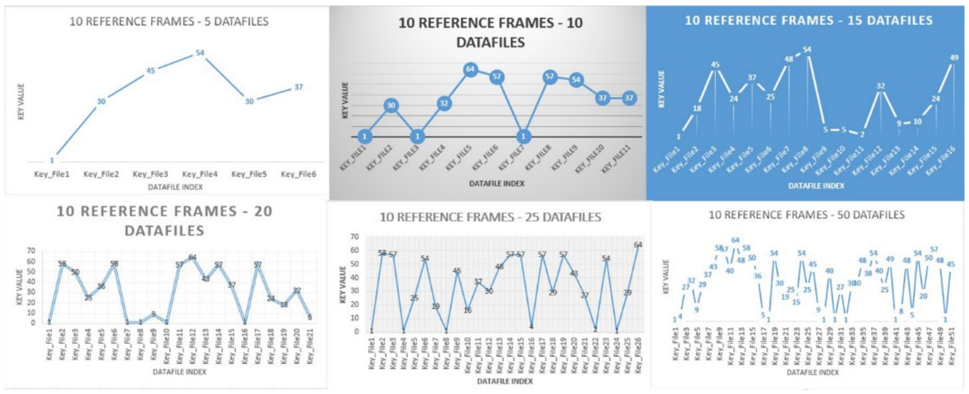


Figure 37 Key value obtained for each Data file with 10 Reference Frames for Simulated Data

Figure 38 shows the graphs with the key values that are obtained for different DataFile count but with fifteen Reference Frames. Each graph (in Figure 96-101) representing the keys generated for different Data File counts are displayed individually in Appendix A.

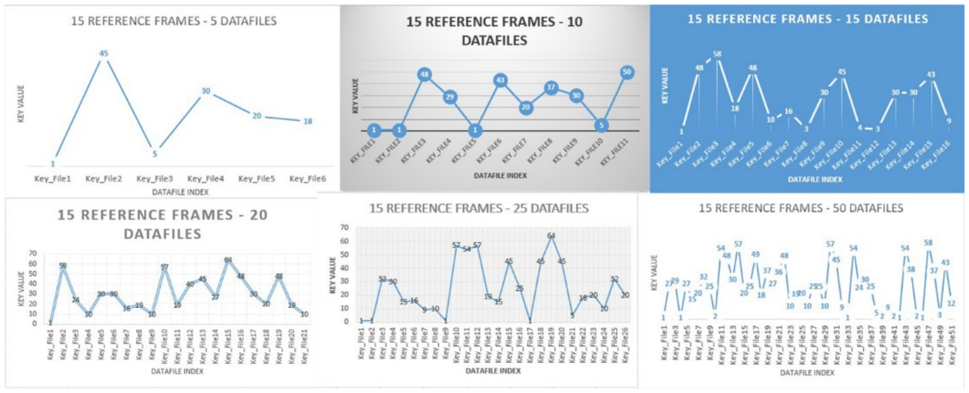


Figure 38 Key value obtained for each Data file with 15 Reference Frames for Simulated Data

Figure 39 shows the graphs with the key values that are obtained for different DataFile count but with twenty Reference Frames. Each graph (in Figure 102-107) representing the keys generated for different Data File counts are displayed individually in Appendix A.

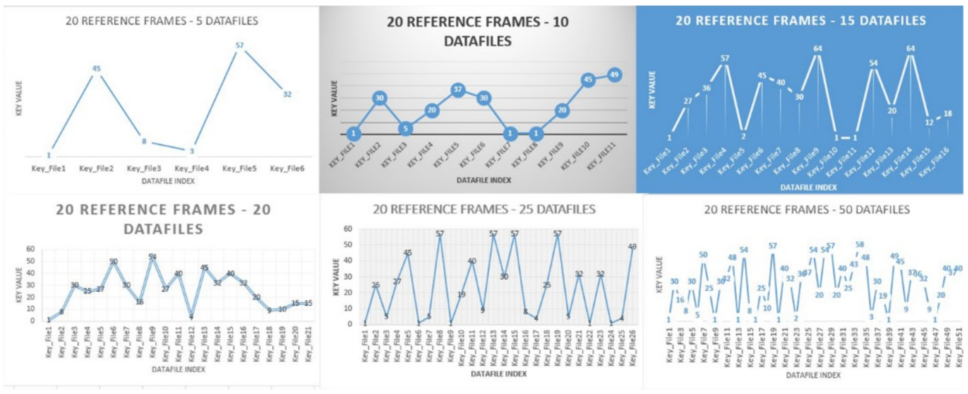


Figure 39 Key value obtained for each Data file with 20 Reference Frames for Simulated Data

Figure 40 shows the graphs with the key values that are obtained for different DataFile count but with twenty-five Reference Frames. Each graph (in Figure 108-113) representing the keys generated for different Data File counts are displayed individually in Appendix A.

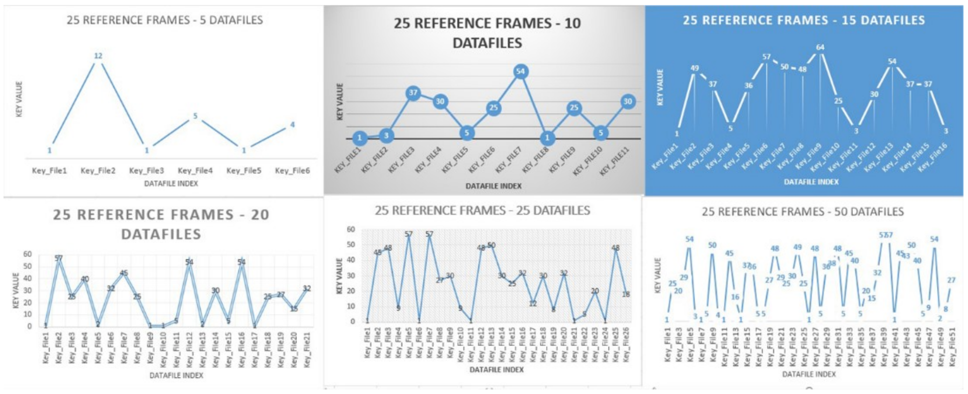


Figure 40 Key value obtained for each Data file with 25 Reference Frames for Simulated Data

As we can see from Figure 35, the execution time remained relatively constant as the Reference Frame count increased. Added to that, Figures 36 to 40 show that there is no pattern which has repeated itself in any of the scenario.

6.3.2 WHEN THE DEVICE IS MOVED 'VERTICALLY'

In this test case, the proposed approach is tested using the data generated by the movement of the mobile device in a vertical direction. This would mimic a situation where the user might be jumping or skipping or hopping in a real time situation. The rest of the interpretation and analysis approach remains the same, as mentioned earlier in Section 6.3.1.

Figure 41 is the image of the Graph which represents the execution time taken for different DataFile count with different Reference Frames.

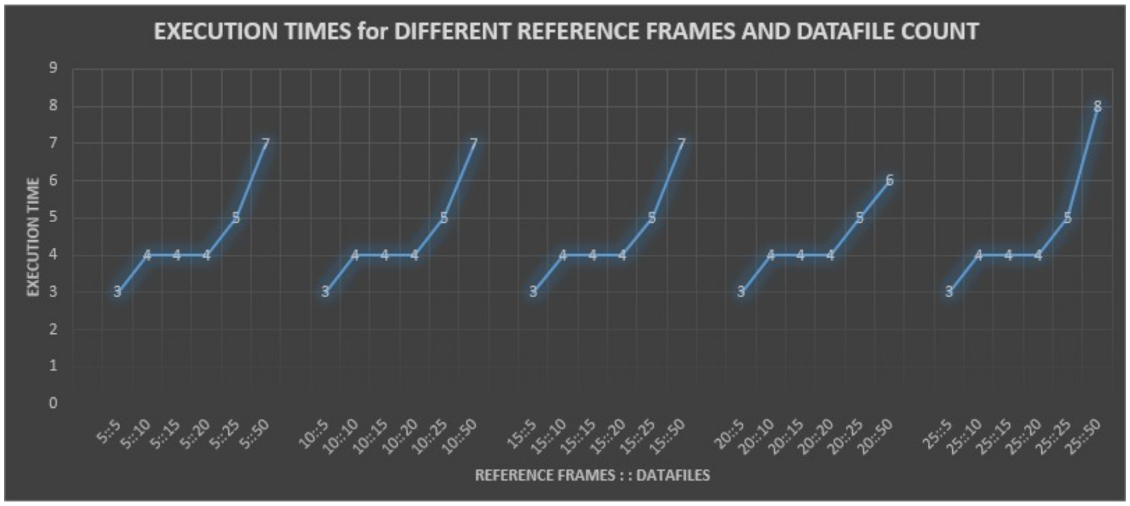


Figure 41 Execution Time vs ReferenceFrames count & DataFile count for Vertical movement of the Device

Figure 42 shows the graphs with the key values that are obtained for different DataFile count but with five Reference Frames. Each graph (in Figure 114-119) representing the keys generated for different Data File counts are displayed individually in Appendix B.

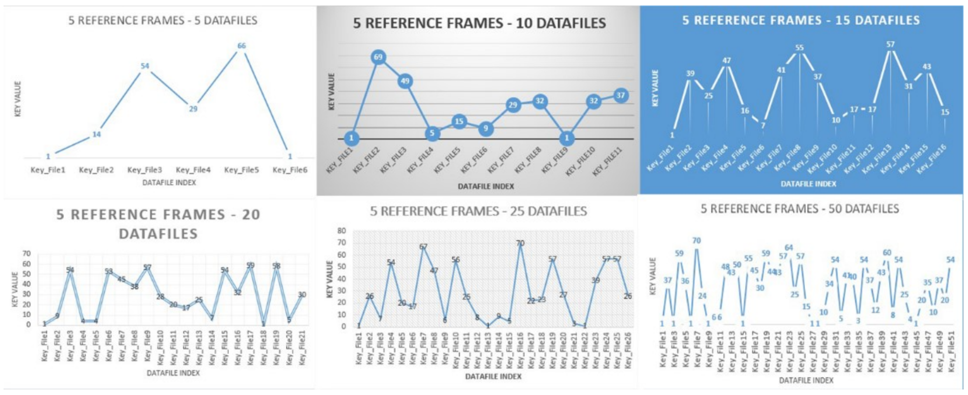


Figure 42 Key value obtained for each Data file with 5 Reference Frames for Vertical movement of the Device

Figure 43 shows the graphs with the key values that are obtained for different DataFile count but with ten Reference Frames. Each graph (in Figure 120-125) representing the keys generated for different Data File counts are displayed individually in Appendix B.

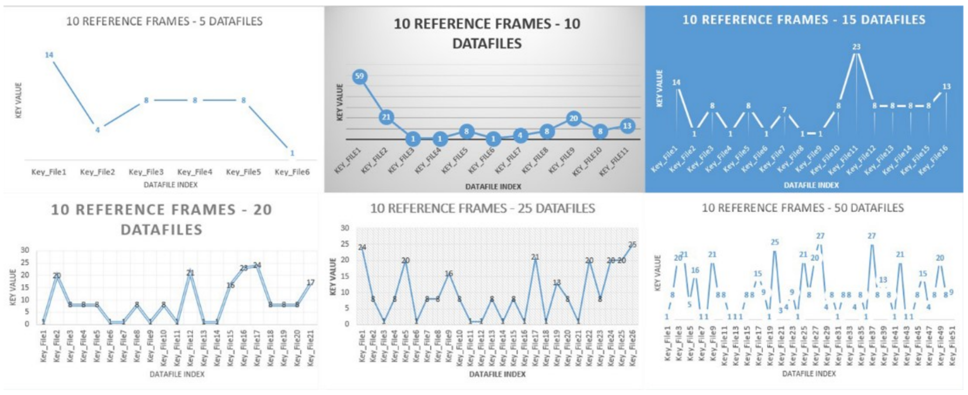


Figure 43 Key value obtained for each Data file with 10 Reference Frames for Vertical movement of the Device

Figure 44 shows the graphs with the key values that are obtained for different DataFile count but with fifteen Reference Frames. Each graph (in Figure 126-131) representing the keys generated for different Data File counts are displayed individually in Appendix B.

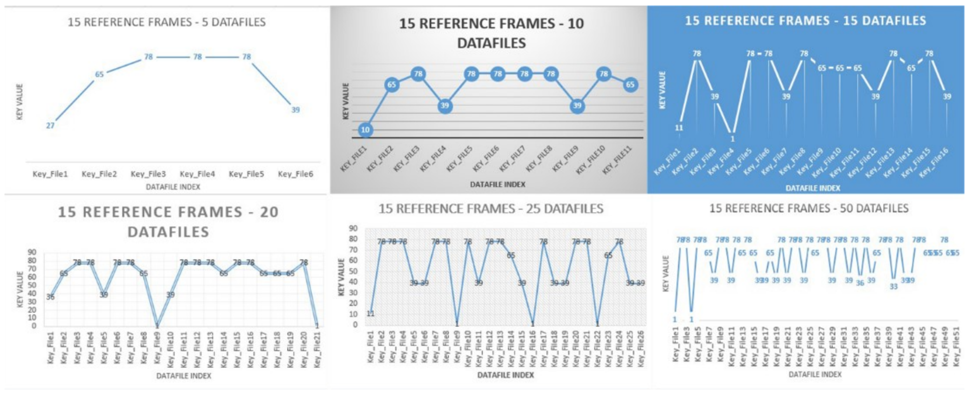


Figure 44 Key value obtained for each Data file with 15 Reference Frames for Vertical movement of the Device

Figure 45 shows the graphs with the key values that are obtained for different DataFile count but with twenty Reference Frames. Each graph (in Figure 132-137) representing the keys generated for different Data File counts are displayed individually in Appendix B.

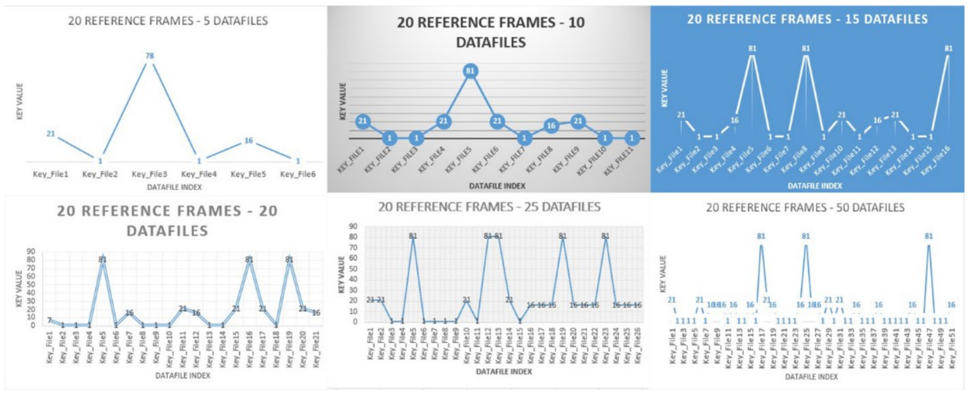


Figure 45 Key value obtained for each Data file with 20 Reference Frames for Vertical movement of the Device

Figure 46 shows the graphs with the key values that are obtained for different DataFile count but with twenty-five Reference Frames. Each graph (in Figure 138-143) representing the keys generated for different Data File counts are displayed individually in Appendix B.

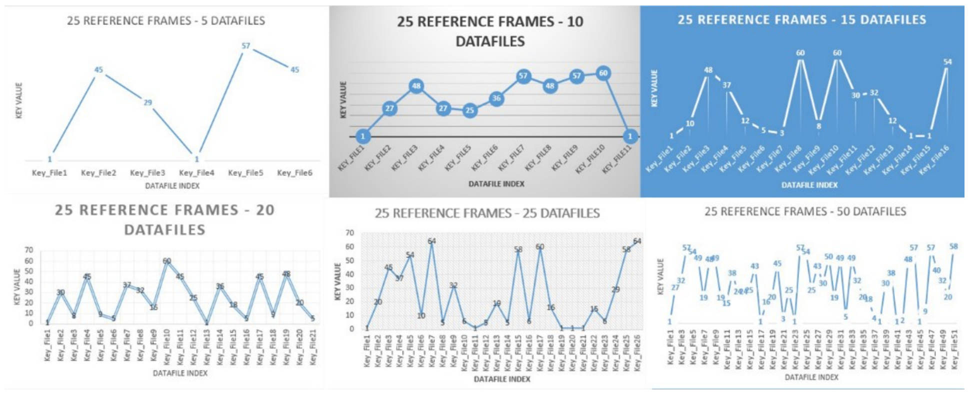


Figure 46 Key value obtained for each Data file with 25 Reference Frames for Vertical movement of the Device

As we can see from Figure 41 the execution time remained relatively constant as the Reference Frame count increased. Added to that, the Figures 42 to 46 show that there is no pattern which has repeated itself in any of the scenario.

6.3.3 WHEN THE DEVICE IS IN 'IDLE' POSITION

In this test case, the proposed approach is tested using the data generated by placing the device in an Idle position i.e. no movement is experienced by the device. This would mimic a situation where the user might be sitting or left the device on a table or sleeping in a real time situation. The rest of the interpretation and analysis remain the same, as mentioned earlier in Section 6.3.1.

Figure 47 is the image of the Graph which represents the execution time taken for different DataFile count with different Reference Frames.

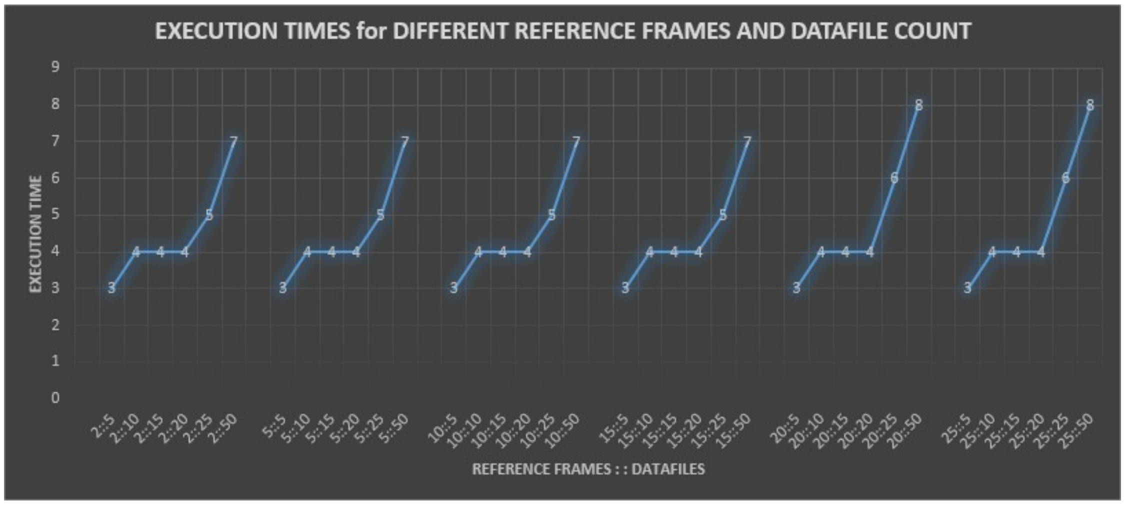


Figure 47 Execution Time vs ReferenceFrames count & DataFile count when the Device is in Idle position

Figure 48 shows the graphs with the key values that are obtained for different DataFile count but with five Reference Frames. Each graph (in Figure 144-149) representing the keys generated for different Data File counts are displayed individually in Appendix C.

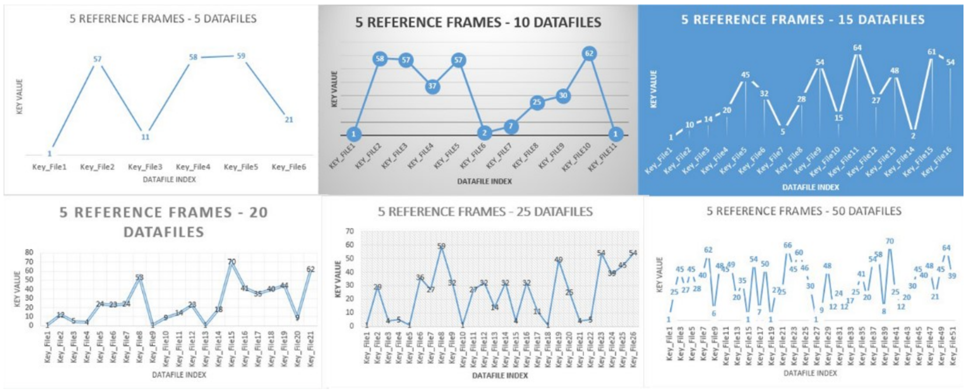


Figure 48 Key value obtained for each Data file with 5 Reference Frames when the Device is in Idle position

Figure 49 shows the graphs with the key values that are obtained for different DataFile count but with ten Reference Frames. Each graph (in Figure 150-155) representing the keys generated for different Data File counts are displayed individually in Appendix C.

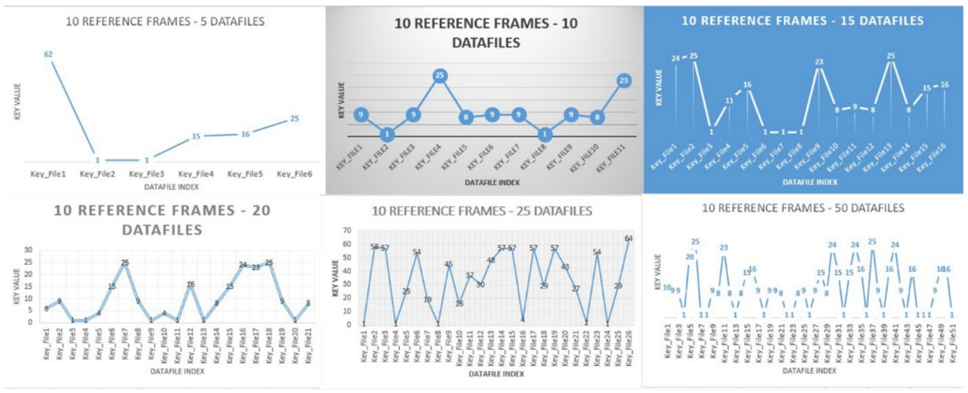


Figure 49 Key value obtained for each Data file with 10 Reference Frames when the Device is in Idle position

Figure 50 shows the graphs with the key values that are obtained for different DataFile count but with fifteen Reference Frames. Each graph (in Figure 156-161) representing the keys generated for different Data File counts are displayed individually in Appendix C.

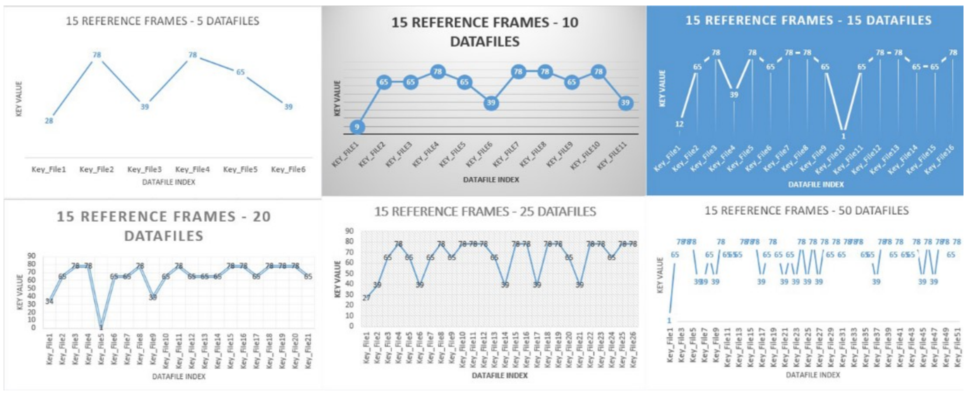


Figure 50 Key value obtained for each Data file with 15 Reference Frames when the Device is in Idle position

Figure 51 shows the graphs with the key values that are obtained for different DataFile count but with twenty Reference Frames. Each graph (in Figure 162-167) representing the keys generated for different Data File counts are displayed individually in Appendix C.

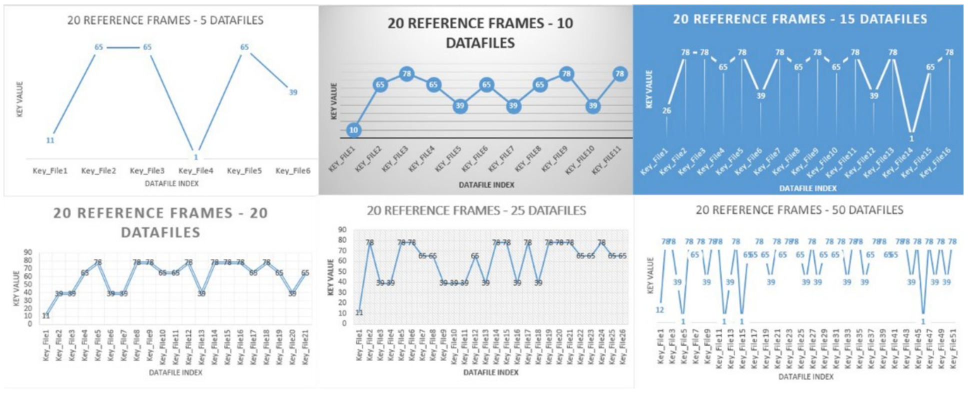


Figure 51 Key value obtained for each Data file with 20 Reference Frames when the Device is in Idle position

Figure 52 shows the graphs with the key values that are obtained for different DataFile count but with twenty-five Reference Frames. Each graph (in Figure 168-173) representing the keys generated for different Data File counts are displayed individually in Appendix C.

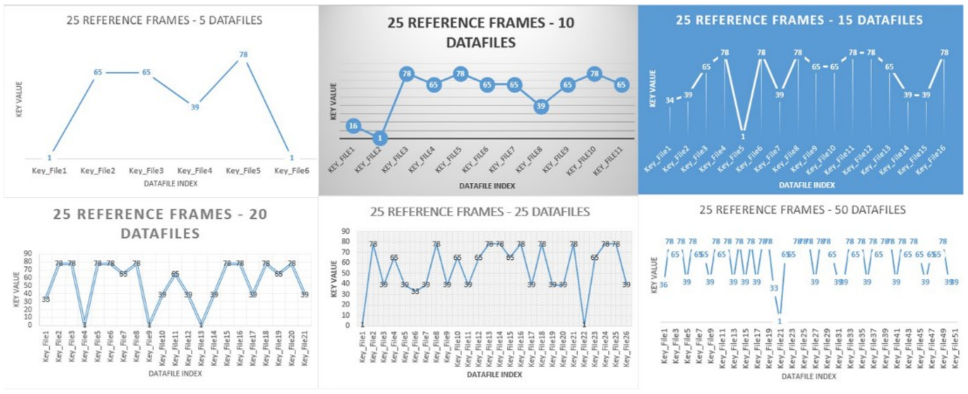


Figure 52 Key value obtained for each Data file with 25 Reference Frames when the Device is in Idle position

As we can see from Figure 47 the execution time remained relatively constant as the Reference Frame count increased. Added to that, the Figures 48 to 52 show that there is no pattern which has repeated itself in any of the scenario.

6.3.4 WHEN THE USER IS 'WALKING' WITH THE DEVICE IN HIS TROUSERS' POCKET

In this test case, the proposed approach is tested using the data generated by placing the device in one of the pockets of the trousers while the researcher is walking. The researcher performed the experiment by walking until all the desired number of files have been created. The rest of the interpretation and analysis remains the same, as mentioned earlier in Section 6.3.1.

Figure 53 is the image of the Graph which represents the execution time taken for different DataFile count with different Reference Frames.

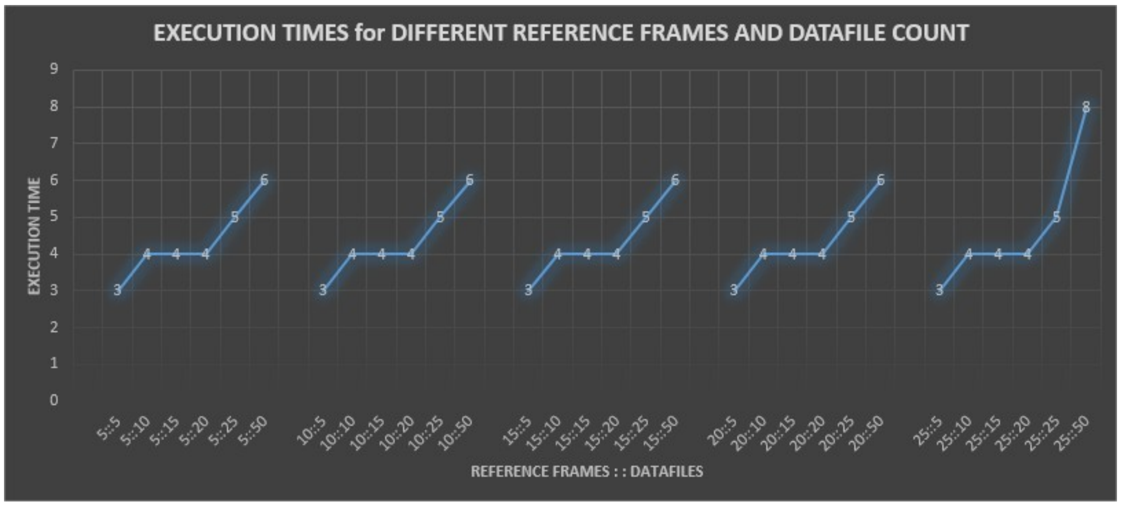


Figure 53 Execution Time vs ReferenceFrames count & DataFile count when the Device is in trouser pocket while walking

Figure 54 shows the graphs with the key values that are obtained for different DataFile count but with five Reference Frames. Each graph (in Figure 174-179) representing the keys generated for different Data File counts are displayed individually in Appendix D.

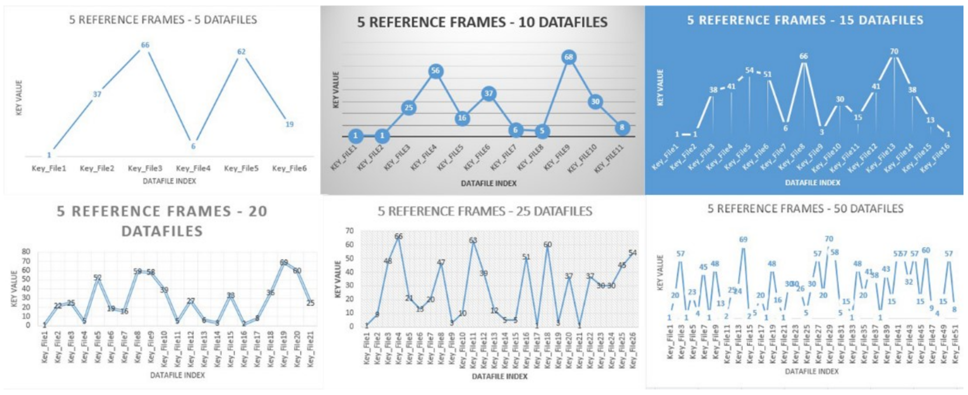


Figure 54 Key value obtained for each Data file with 5 Reference Frames when the Device is in trouser pocket while walking

Figure 52 shows the graphs with the key values that are obtained for different DataFile count but with ten Reference Frames. Each graph (in Figure 180-185) representing the keys generated for different Data File counts are displayed individually in Appendix D.

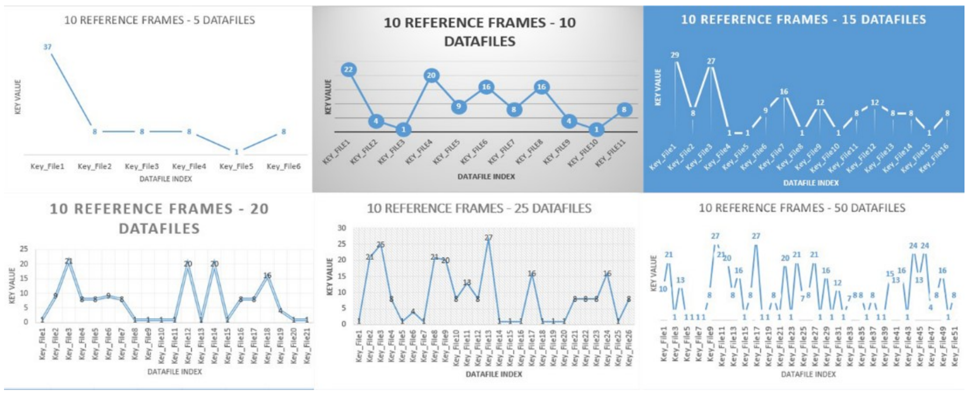


Figure 55 Key value obtained for each Data file with 10 Reference Frames when the Device is in trouser pocket while walking

Figure 56 shows the graphs with the key values that are obtained for different DataFile count but with fifteen Reference Frames. Each graph (in Figure 186-191) representing the keys generated for different Data File counts are displayed individually in Appendix D.

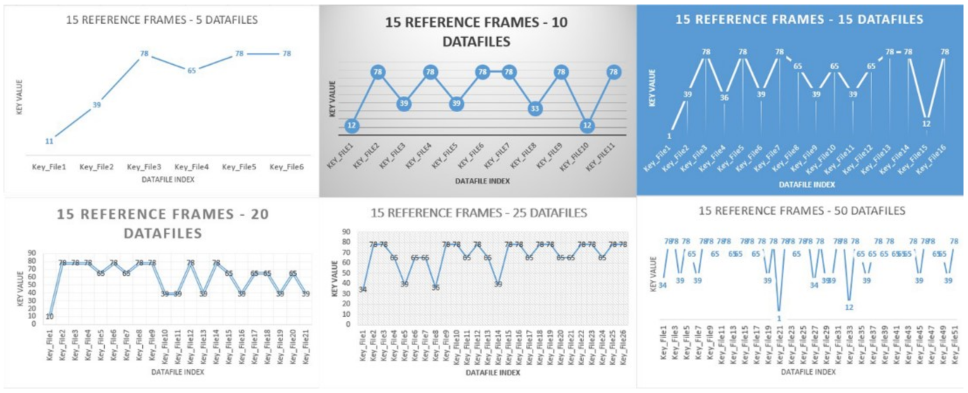


Figure 56 Key value obtained for each Data file with 15 Reference Frames when the Device is in trouser pocket while walking

Figure 57 shows the graphs with the key values that are obtained for different DataFile count but with twenty Reference Frames. Each graph (in Figure 192-197) representing the keys generated for different Data File counts are displayed individually in Appendix D.

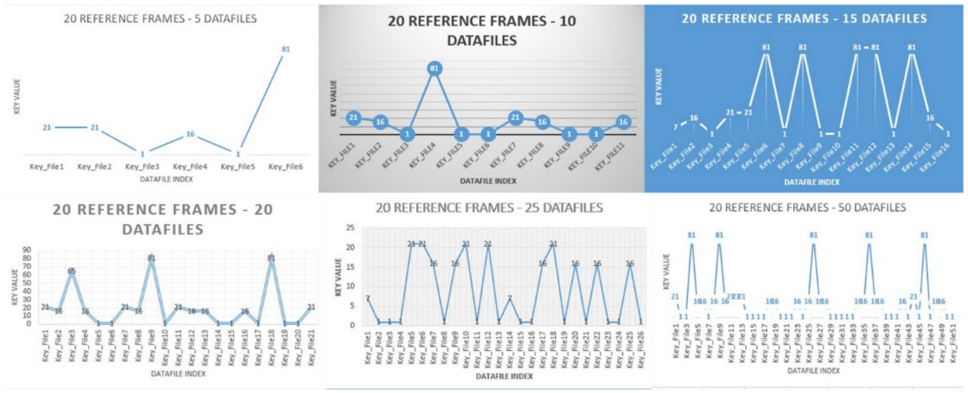


Figure 57 Key value obtained for each Data file with 20 Reference Frames when the Device is in trouser pocket while walking

Figure 58 shows the graphs with the key values that are obtained for different DataFile count but with twenty-five Reference Frames. Each graph (in Figure 198-203) representing the keys generated for different Data File counts are displayed individually in Appendix D.

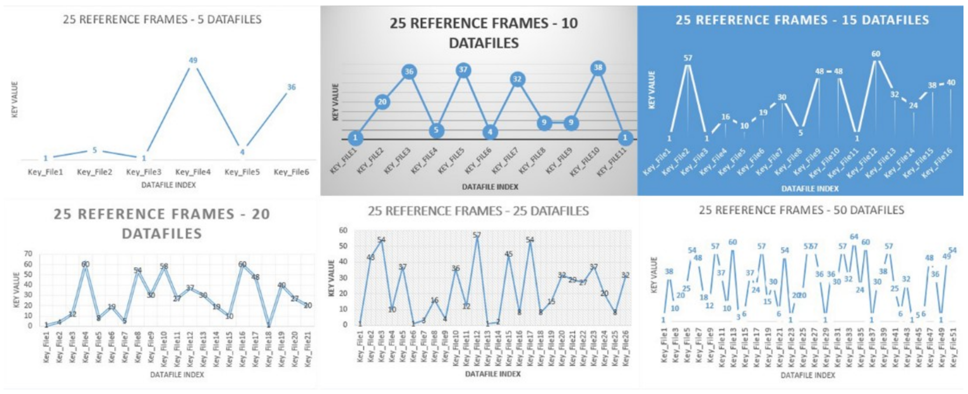


Figure 58 Key value obtained for each Data file with 25 Reference Frames when the Device is in trouser pocket while walking

As we can see from Figure 53 the execution time remained relatively constant as the Reference Frame count increased. Added to that, the Figures 54 to 58 show that there is no pattern which has repeated itself in any of the scenario.

6.3.5 WHEN THE USER IS INSIDE AN 'ELEVATOR' WITH THE DEVICE IN HIS TROUSERS' POCKET

In this test case, the proposed approach is tested using the data generated by placing the device in one of the pockets of the trousers while the researcher is inside an elevator. The researcher performed the experiment by remaining inside an elevator and moving up/down to different floors at random until all the desired number of files have been created. The rest of the interpretation and analysis remains the same, as mentioned earlier in Section 6.3.1.

Since the Execution time for different DataFile count remained the same in all of the above discussed test cases, for the following test cases, the experiment is tested for different Reference Frame count, having the DataFile count fixed to five.

Figures 59 shows the graphs with the key values that are obtained for different DataFiles with five Reference Frames.

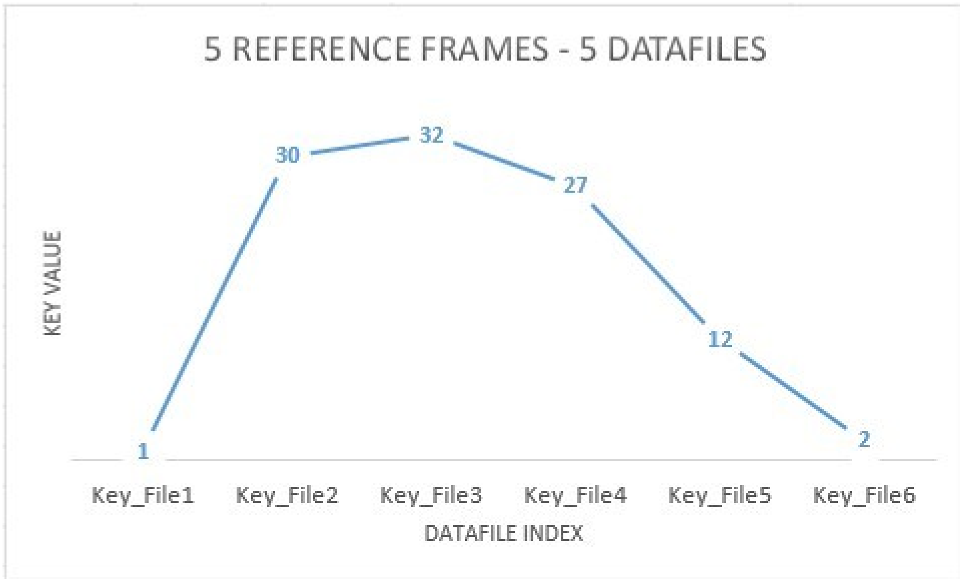


Figure 59 Key value obtained with 5 Reference Frames when the Device is in trouser pocket while in elevator

Figure 60 shows the graphs with the key values that are obtained for different DataFiles with ten Reference Frames.

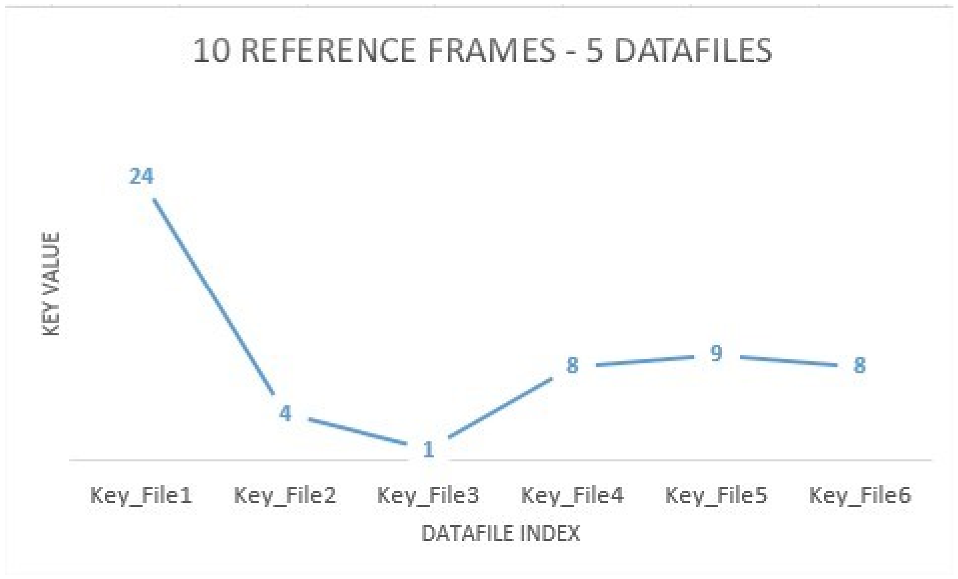


Figure 60 Key value obtained with 10 Reference Frames when the Device is in trouser pocket while in elevator

Figure 61 shows the graphs with the key values that are obtained for different DataFiles with fifteen Reference Frames.

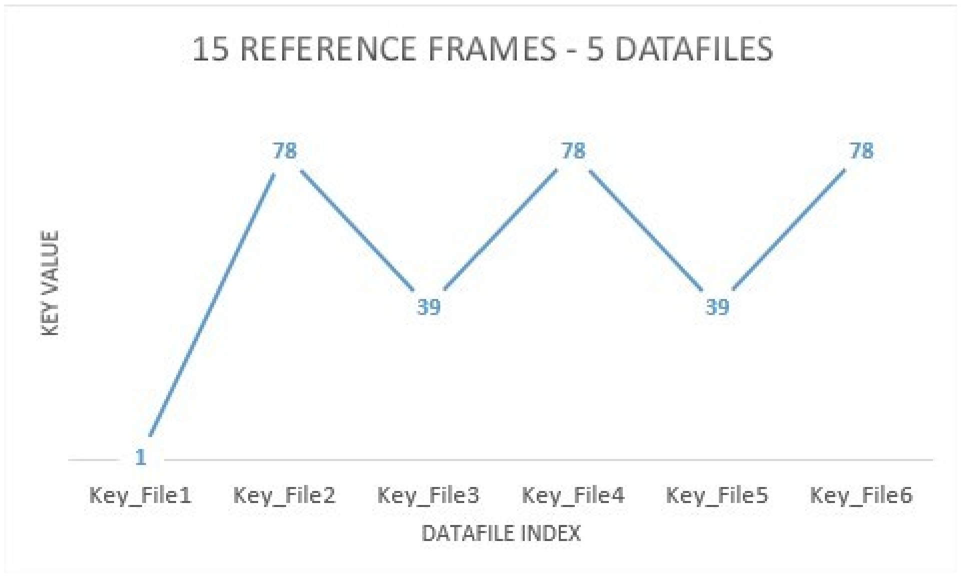


Figure 61 Key value obtained with 15 Reference Frames when the Device is in trouser pocket while in elevator

Figure 62 shows the graphs with the key values that are obtained for different DataFiles with twenty Reference Frames.

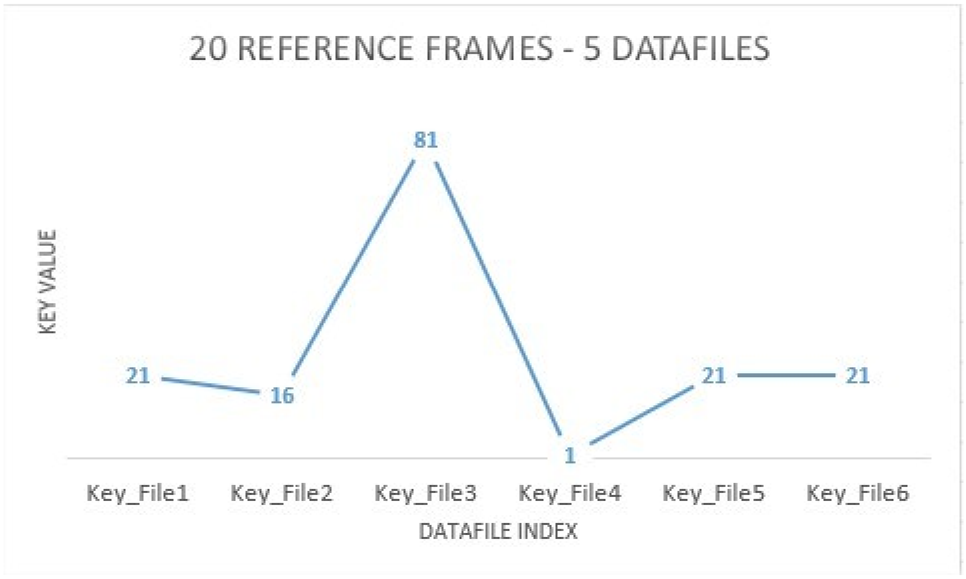


Figure 62 Key value obtained 20 Reference Frames when the Device is in trouser pocket while in elevator

Figure 63 shows the graphs with the key values that are obtained for different DataFiles with twenty-five Reference Frames.

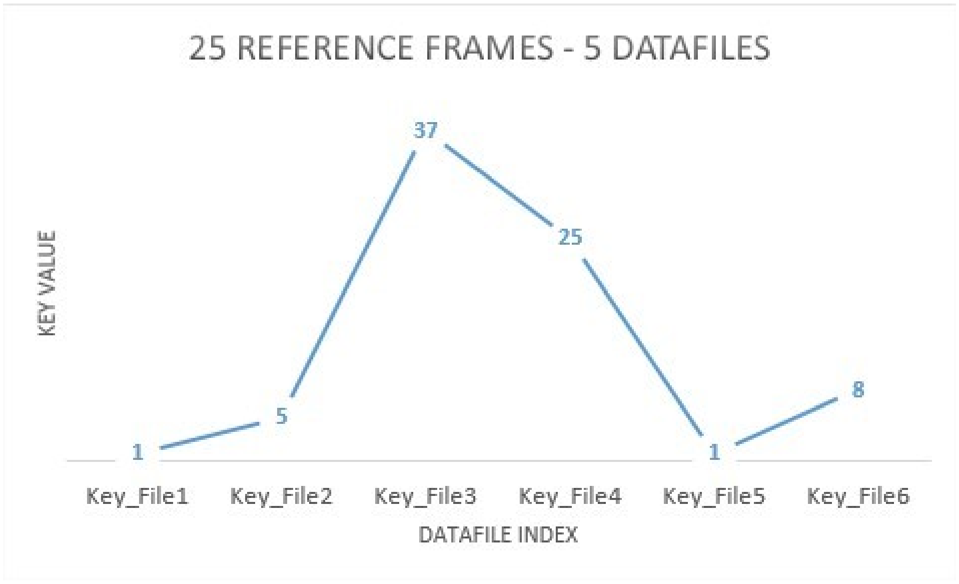


Figure 63 Key value obtained with 25 Reference Frames.

From the Figures 59 to 63 it is clear that there is no pattern which has repeated itself in any of the scenario.

6.3.6 WHEN THE USER IS MOVING ON THE 'STAIRS' WITH THE DEVICE IN HIS TROUSERS' POCKET

In this test case, the proposed approach is tested using the data generated by placing the device in one of the pockets of the trousers while the researcher is climbing up the stairs and then going down the stairs. The researcher performed the experiment by moving up and down to different floors at random until all the desired number of files have been created. The rest of the interpretation and analysis approach remains the same, as mentioned earlier in Section 6.3.1.

This experiment is tested for different Reference Frame count, having the DataFile count fixed to five.

Figure shows the graphs with the key values that are obtained for different DataFiles with five Reference Frames.

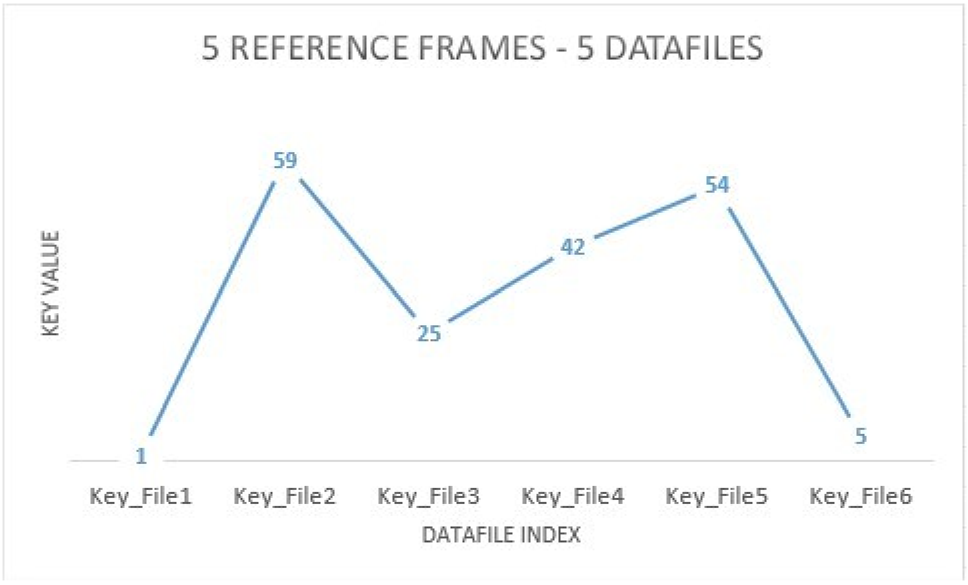


Figure 64 Key value obtained with 5 Reference Frames when the Device is in trouser pocket while walking in stairs

Figure 65 shows the graphs with the key values that are obtained for different DataFiles with ten Reference Frames.

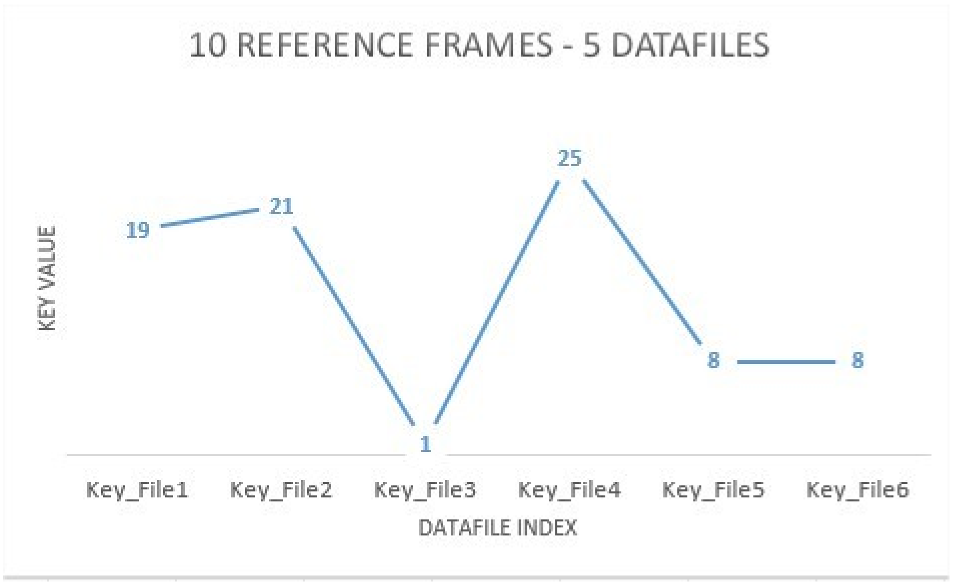


Figure 65 Key value obtained with 10 Reference Frames when the Device is in trouser pocket while walking in stairs

Figure 66 shows the graphs with the key values that are obtained for different DataFiles with fifteen Reference Frames.

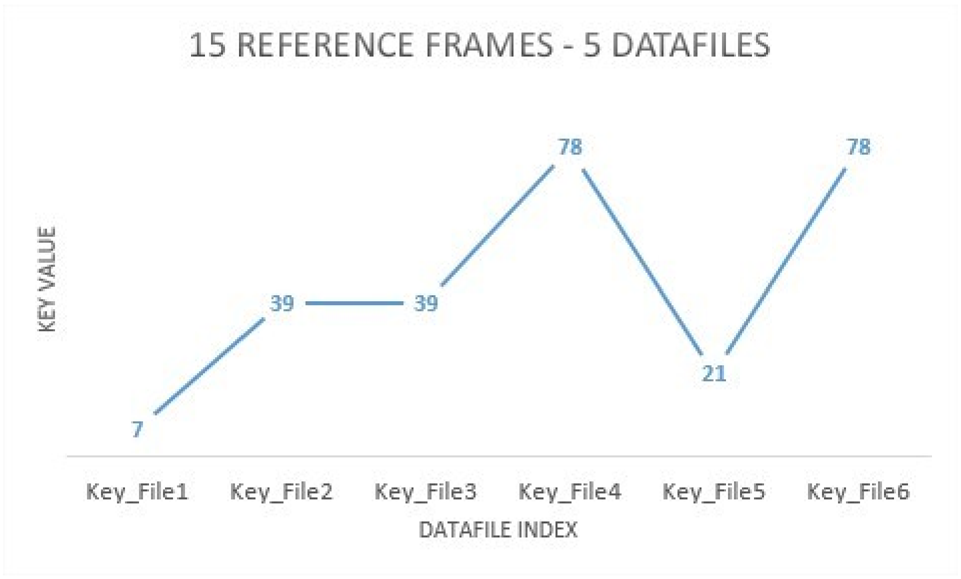


Figure 66 Key value obtained with 15 Reference Frames when the Device is in trouser pocket while walking in stairs

Figure 67 shows the graphs with the key values that are obtained for different DataFiles with twenty Reference Frames.

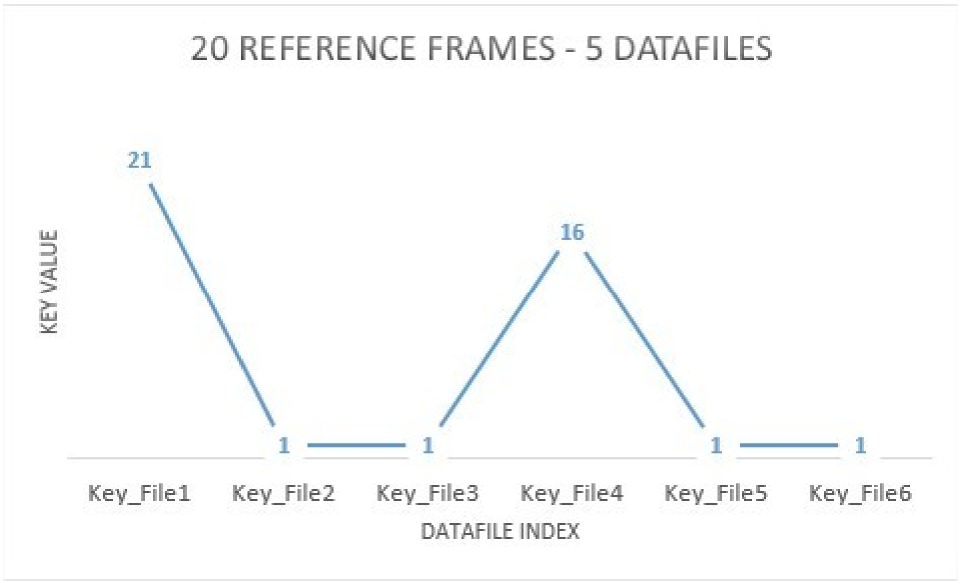


Figure 67 Key value obtained 20 Reference Frames when the Device is in trouser pocket while walking in stairs

The following figure shows the graphs with the key values that are obtained for different DataFiles with twenty-five Reference Frames.

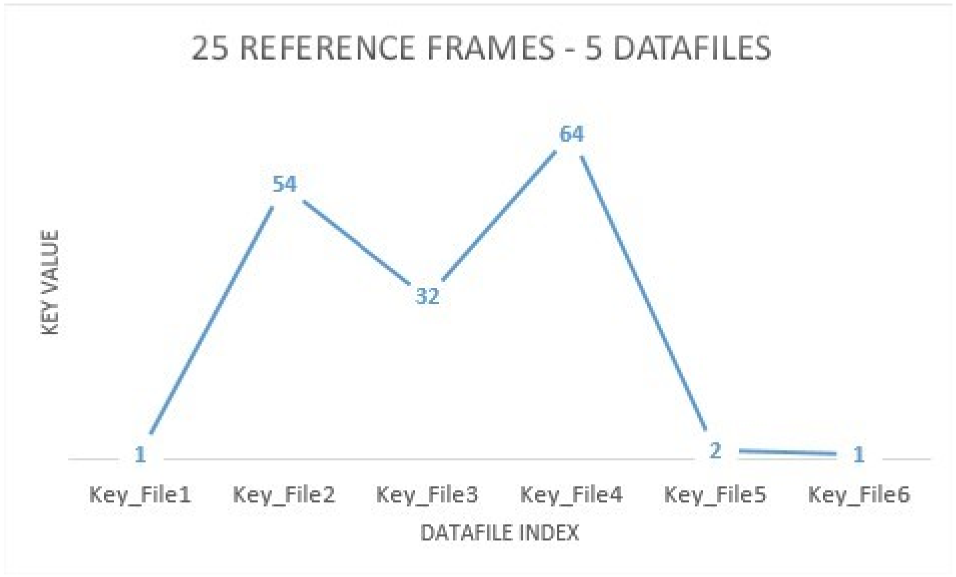


Figure 68 Key value obtained with 25 Reference Frames when the Device is in trouser pocket while walking in stairs

From the Figures 64 to 68, it is clear that there is no pattern which has repeated itself in any of the scenario.

6.3.7 WHEN THE USER IS WITHIN A 'VEHICLE' WITH THE DEVICE IN HIS TROUSERS' POCKET

In this test case, the proposed approach is tested using the data generated by placing the device in one of the pockets of the trousers while the researcher is inside a moving vehicle such as a bus. The researcher performed the experiment by remaining inside the bus, which halted at different stops at random, until all the desired number of files have been created. The rest of the interpretation and analysis remains the same, as mentioned earlier in Section 6.3.1.

The experiment is tested for different Reference Frame count, having the DataFile count fixed to five.

Figure 69 shows the graphs with the key values that are obtained for different DataFiles with five Reference Frames.

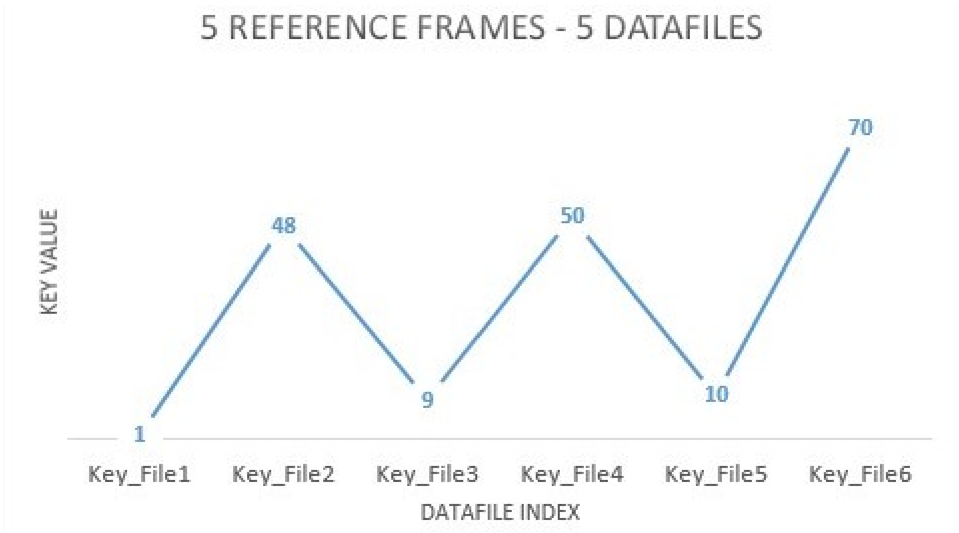


Figure 69 Key value obtained with 5 Reference Frames when the Device is in trouser pocket while moving in vehicle

Figure 70 shows the graphs with the key values that are obtained for different DataFiles with ten Reference Frames.

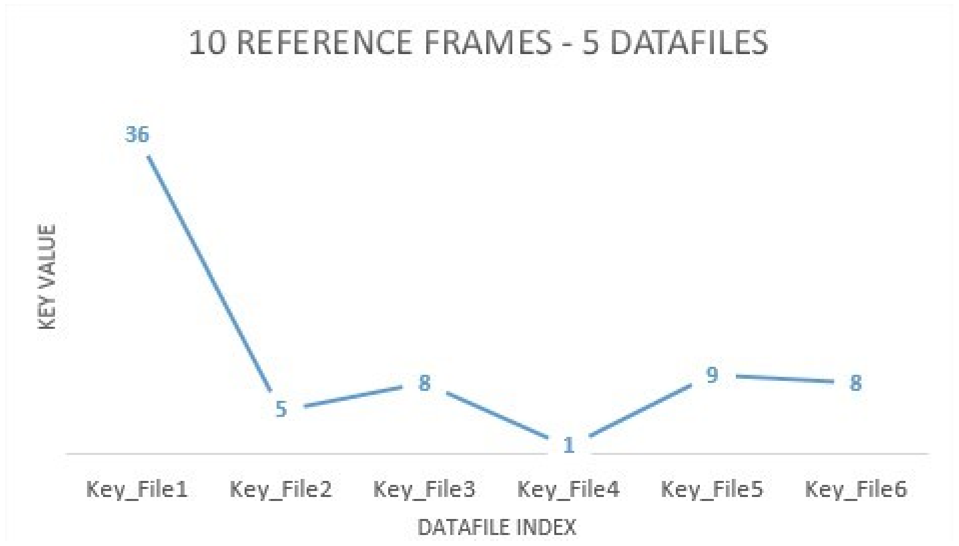


Figure 70 Key value obtained with 10 Reference Frames when the Device is in trouser pocket while moving in vehicle

Figure 71 shows the graphs with the key values that are obtained for different DataFiles with fifteen Reference Frames.

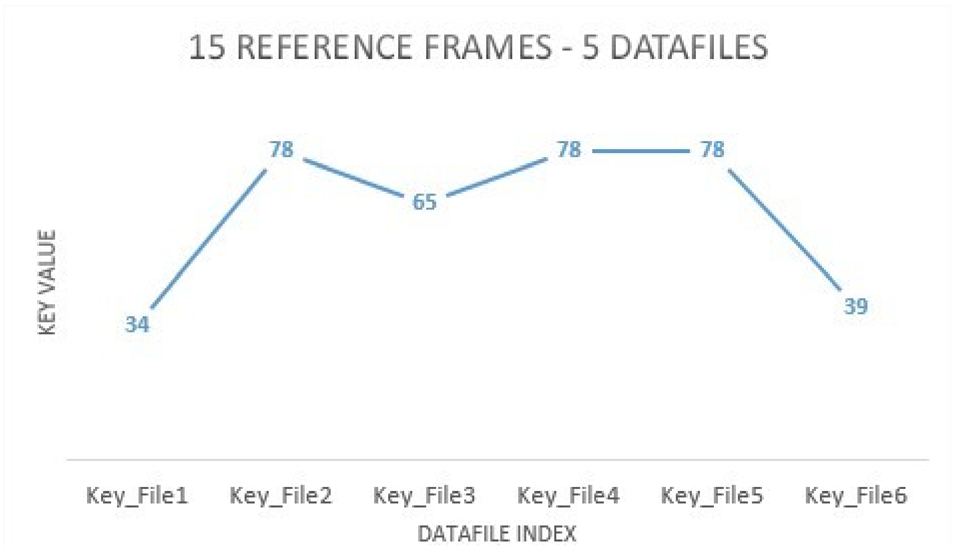


Figure 71 Key value obtained with 15 Reference Frames when the Device is in trouser pocket while moving in vehicle

Figure 72 shows the graphs with the key values that are obtained for different DataFiles with twenty Reference Frames.

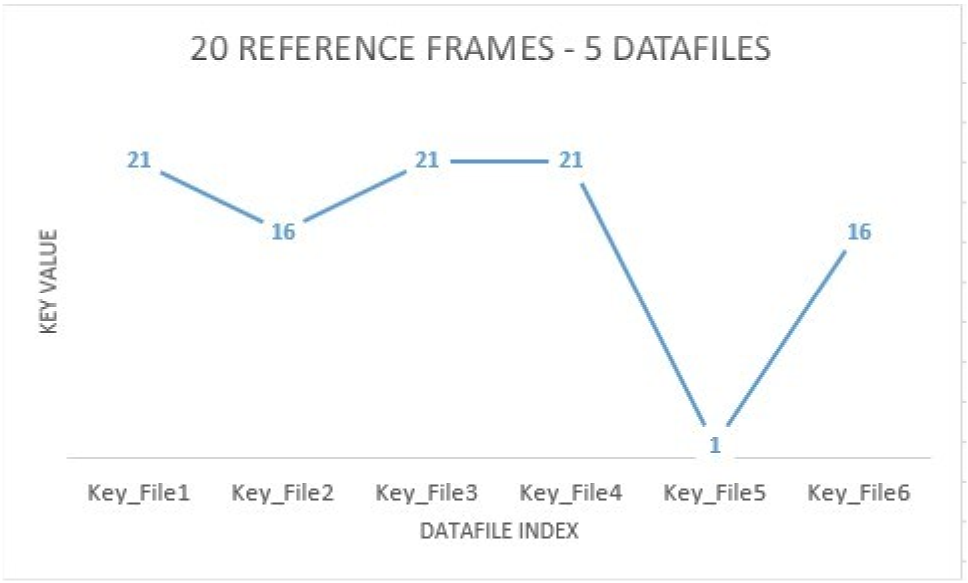


Figure 72 Key value obtained 20 Reference Frames when the Device is in trouser pocket while moving in vehicle

Figure 73 shows the graphs with the key values that are obtained for different DataFiles with twenty-five Reference Frames.

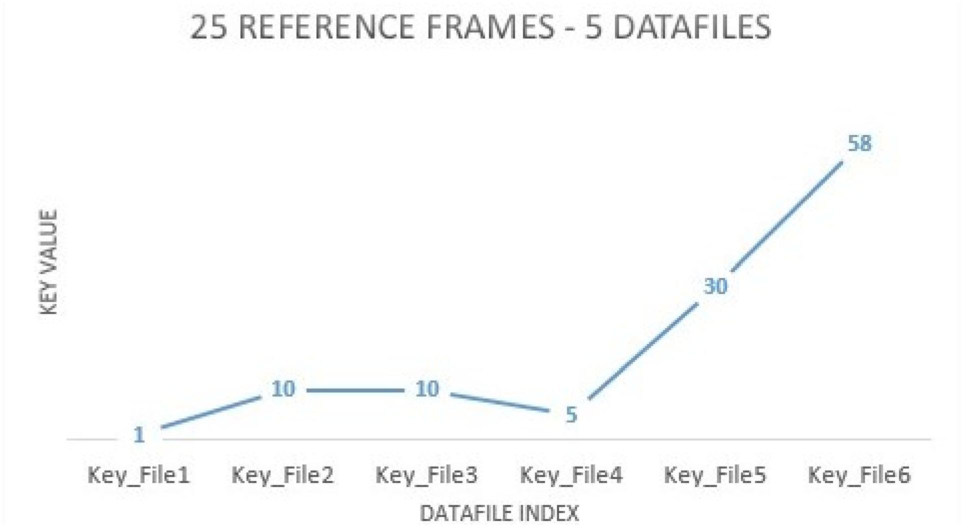


Figure 73 Key value obtained with 25 Reference Frames when the Device is in trouser pocket while moving in vehicle

From the Figures 69 until 73, it is clear that there is no pattern which has repeated itself in any of the scenario.

6.3.8 INFERENCES FROM THE TEST CASES

In the experiments performed for test cases described in Section 6.3.1 until 6.3.4, the time required for the entire proposed approach to be processed and the communication to be completed, is recorded and plotted into a graph. In each test case, each set of Reference Frames count was applied to each set of DataFiles count. The keys thus generated are also stored and analyzed to check for randomness.

From our observation, the execution time depended on the number of files being encrypted and transmitted over the wire. The number of reference frames did not influence the execution time. The execution time increased with the increase in DataFiles count. But for the same DataFiles count, with the increase in the Reference Frames count, the execution time remained almost same. This proves that the number of Reference Frames does not affect the execution time but from our algorithm it does increase the choices from which the values can be chosen from. In short, if there are 2 Reference Frames with 5 Fields each, then a value ‘x’ can be one among the ten {2*5} available values. But if we are working with 10 Reference Frames with 5 Fields each, then the value ‘x’ can be one amongst the fifty {10*5} available values. The latter scenario has lesser chances of predicting the value of ‘x’ in comparison to the former scenario. Hence there is no relation between the number of Reference Frames and the number of Data Files. Any number of Reference Frames can be considered for transferring any number of Data Files. There might be an increase in the execution time due to the increase in the search space while key generation but this would have no observable effect on the data files that are being transferred. We started observing the Execution time to check if there is any noticeable effect of the increase in the Reference Frames w.r.t. the data files count. Apart from that, there is no role what so ever of the execution time in this thesis. Based on our observation, we have not noticed any drastic hike in the execution time as the number of Reference Frames increased until 25. Further testing is required in order to observe or comment on the execution time consumption for Reference Frame counts beyond 25.

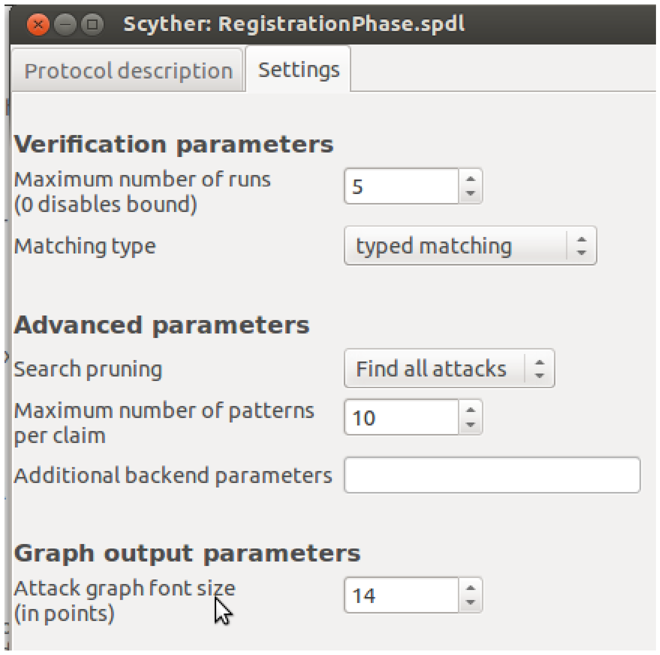
The set of graphs displaying the Keys’ values in each test case for each ‘Reference Frame count’ - ‘DataFile count’ combination, also present interesting results. None of the graph pattern is observed to repeat itself in any other graph yielded by any test case. Each graph’s pattern is unique and is not redundant. The values of the keys are seen to repeat itself in some scenarios. But the file number or the time at which the key is repeated, are also observed to be random. The repetition of the keys is due to an implementation limitation. The ‘BigInteger’ structure was not able to store the entire size of the generated intermediate values i.e. if the length of ‘p’, ‘g’ and ‘Ta’ was 7 digits long, then for the value generated after the calculation of key be beyond the bounds of the BigInteger capacity (7 digits to the power of 7 digits is observed to go beyond the BigInteger size). Thus in order to handle the limitation, the proposed work was implemented by using only 2 digits for each.

Hence in short, the keys generated were observed to be random and un-predictive in nature for all of the test cases.

6.4 Evaluation using Scyther

Scyther is a protocol analyzer tool which verifies the security aspect of the protocol proposed. It displays all the possible attack which can be launched on the protocol. Scyther assumes that the adversary has full access to the contents of the communicating channel [62][54].

The Scyther environment for the testing of the proposed approach is set to find all attacks that are possible. The maximum number of runs is set to the default value which is five. ‘Claim’ is used in Scyther to check if the transmitted data/datum is secure or not. In this environment, ten pattern counts per claim, which is the default value, is used to validate the claim. The following figure shows the configuration of the Scyther settings.



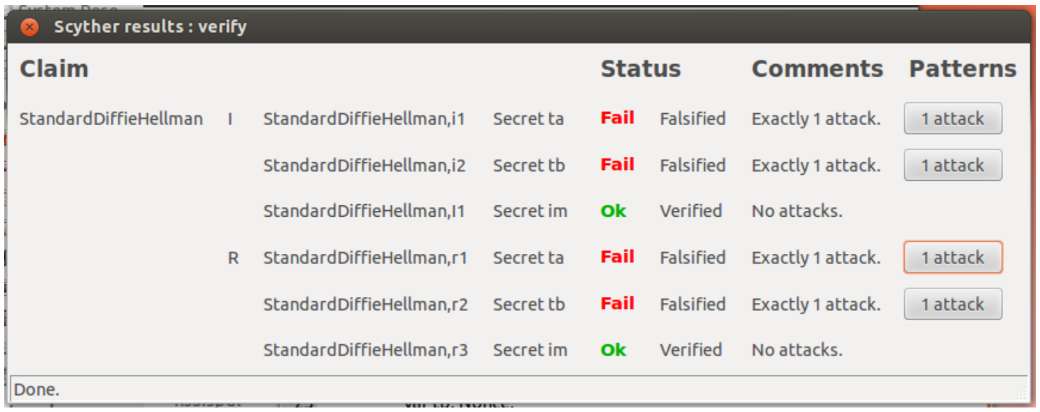
Verification parameters
Maximum number of runs (0 disables bound): 5
Matching type: typed matching

Advanced parameters
Search pruning: Find all attacks
Maximum number of patterns per claim: 10
Additional backend parameters:

Graph output parameters
Attack graph font size (in points): 14

Figure 74 Scyther configuration settings.

We tested the Standard Diffie Hellman in Scyther and figures 75-80 illustrate how MITM can occur between two communicating entities.

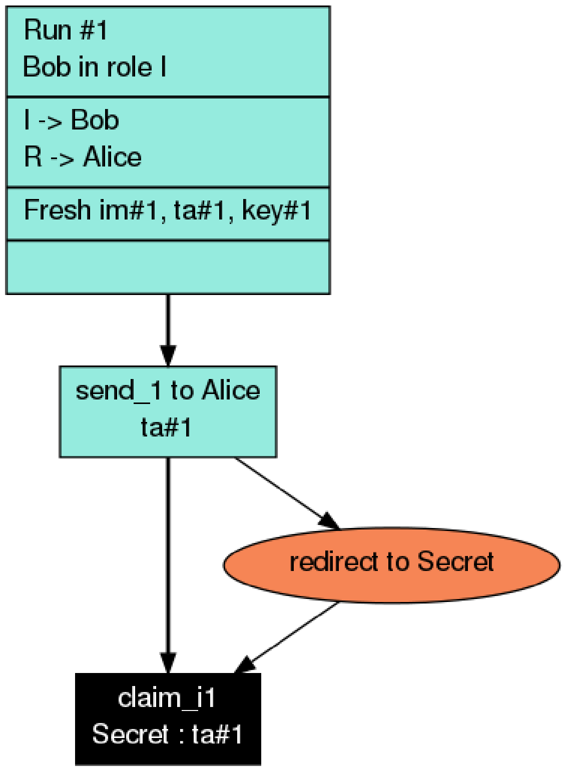


Scyther results : verify

Claim				Status		Comments	Patterns
StandardDiffieHellman	I	StandardDiffieHellman,i1	Secret ta	Fail	Falsified	Exactly 1 attack.	1 attack
		StandardDiffieHellman,i2	Secret tb	Fail	Falsified	Exactly 1 attack.	1 attack
		StandardDiffieHellman,I1	Secret im	Ok	Verified	No attacks.	
	R	StandardDiffieHellman,r1	Secret ta	Fail	Falsified	Exactly 1 attack.	1 attack
		StandardDiffieHellman,r2	Secret tb	Fail	Falsified	Exactly 1 attack.	1 attack
		StandardDiffieHellman,r3	Secret im	Ok	Verified	No attacks.	

Done.

Figure 75 Attacks on Diffie Hellman



[Id 1] Protocol StandardDiffieHellman, role I, claim type Secret

Figure 76 ‘Ta’ compromised

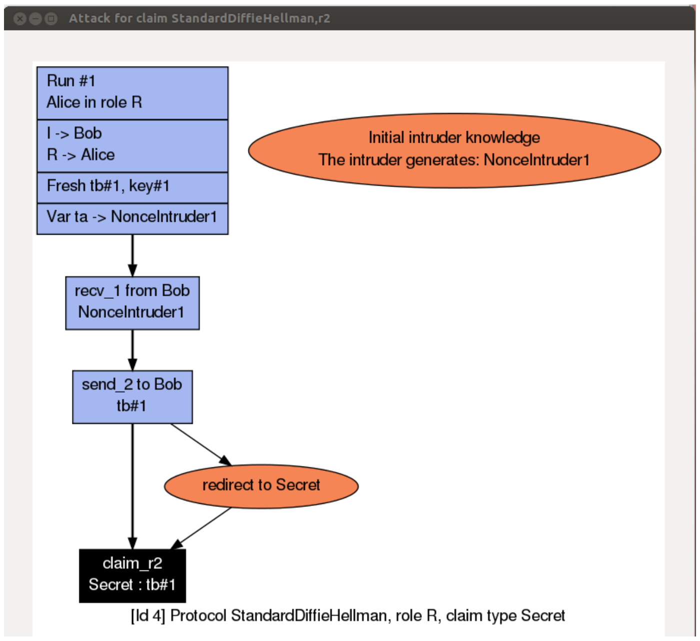
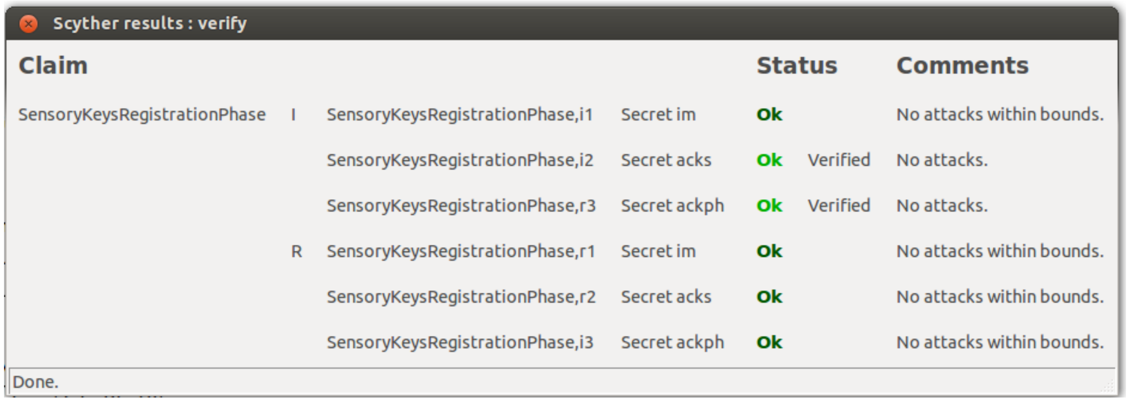

Attack for claim StandardDiffieHellman,r2
Run #1
Alice in role R
I -> Bob
R -> Alice
Fresh tb#1, key#1
Var ta -> NonceIntruder1
Initial intruder knowledge
The intruder generates: NonceIntruder1
recv_1 from Bob
NonceIntruder1
send_2 to Bob
tb#1
redirect to Secret
claim_r2
Secret : tb#1
[Id 4] Protocol StandardDiffieHellman, role R, claim type Secret

Figure 77 ‘Tb’ compromised

We observed that an intruder can invade into the communication channel and compromise the contents (Figure 75). Since there is no default authentication included in the Diffie Hellman, hence any adversary can alter the content of the data packet. As seen in Figure 76 and Figure 76, the ‘Ta’ is compromised. It can be altered and then forwarded to the other entity which leads to the MITM as described in Section 2.5.1

For the Registration Phase, Scyther states that all the terms which are exchanged over the communication channel in the proposed approach, are secure. Figure 78 shows the output of Scyther for the Registration Phase.

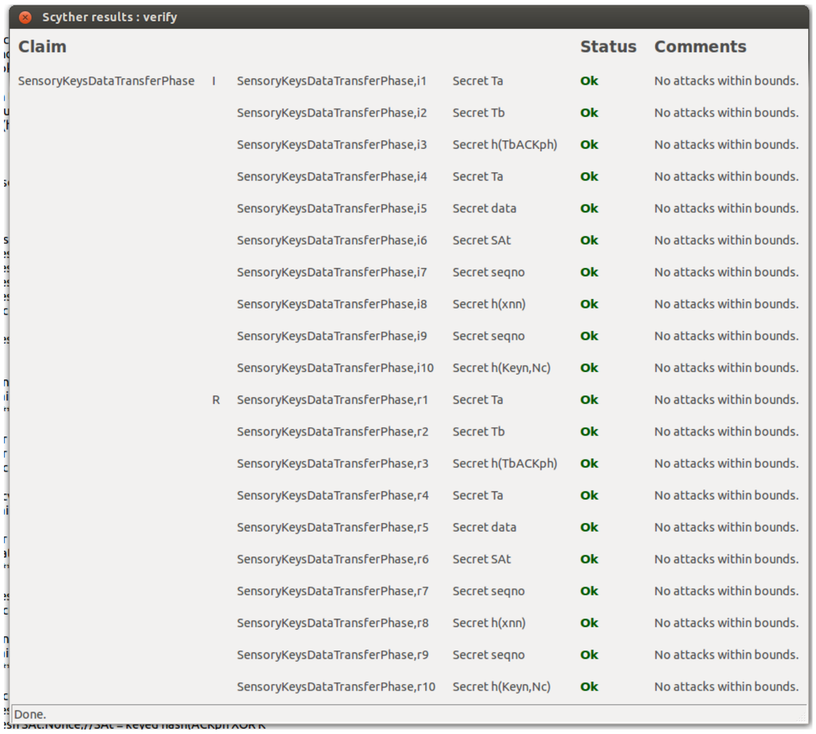

Scyther results : verify

Claim				Status		Comments
SensoryKeysRegistrationPhase	I	SensoryKeysRegistrationPhase,i1	Secret im	Ok		No attacks within bounds.
		SensoryKeysRegistrationPhase,i2	Secret acks	Ok	Verified	No attacks.
		SensoryKeysRegistrationPhase,r3	Secret ackph	Ok	Verified	No attacks.
	R	SensoryKeysRegistrationPhase,r1	Secret im	Ok		No attacks within bounds.
		SensoryKeysRegistrationPhase,r2	Secret acks	Ok		No attacks within bounds.
		SensoryKeysRegistrationPhase,i3	Secret ackph	Ok		No attacks within bounds.

Done.

Figure 78 Scyther output for Registration Phase.

For the Data Transfer Phase, Scyther states that all the terms which are exchanged over the communication channel in the proposed approach, are secure. Figure 79 shows the obtained output of Scyther for the Data Transfer Phase.



Scyther results : verify

Claim				Status	Comments
SensoryKeysDataTransferPhase	I	SensoryKeysDataTransferPhase,i1	Secret Ta	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,i2	Secret Tb	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,i3	Secret h(TbACKph)	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,i4	Secret Ta	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,i5	Secret data	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,i6	Secret SAt	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,i7	Secret seqno	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,i8	Secret h(xnn)	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,i9	Secret seqno	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,i10	Secret h(Keyn,Nc)	Ok	No attacks within bounds.
	R	SensoryKeysDataTransferPhase,r1	Secret Ta	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,r2	Secret Tb	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,r3	Secret h(TbACKph)	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,r4	Secret Ta	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,r5	Secret data	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,r6	Secret SAt	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,r7	Secret seqno	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,r8	Secret h(xnn)	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,r9	Secret seqno	Ok	No attacks within bounds.
		SensoryKeysDataTransferPhase,r10	Secret h(Keyn,Nc)	Ok	No attacks within bounds.

Done.

Figure 79 Scyther output for Data Transfer Phase.

Hence Figure 75-77 present us the possibility of MITM in standard Diffie Hellman and Figures 78 & 79 show that no attacks are possible within bounds in our proposed approach. As far as Scyther is concerned, no intruder can eavesdrop or modify the contents of the data packet in the proposed approach. In essence, our proposed technique is able to preserve Confidentiality, Integrity and Authentication factors of the Security Analysis.

6.5 PERFORMANCE ANALYSIS

We implemented the basic Diffie Hellman algorithm in a java program in a similar client-server architecture as used for the proposed approach. We then retrieved the RAM memory usage, the CPU process time and the CPU up-time for both of the programs under similar conditions (by ‘conditions’, we refer to the same set of 5 Reference Frames and same count of Data Files). Herein RAM memory usage means the amount of RAM memory used for the execution of the program, CPU up-time is the duration for which the CPU was actively used for the program and CPU process time is the amount of time CPU was used for processing the instructions available in the program. The recorded values are plotted onto a graph. Figure 80 is the graph obtained for the CPU up-time, Figure 81 is the graph for the CPU process time and Figure 82 is the graph for the RAM usage.

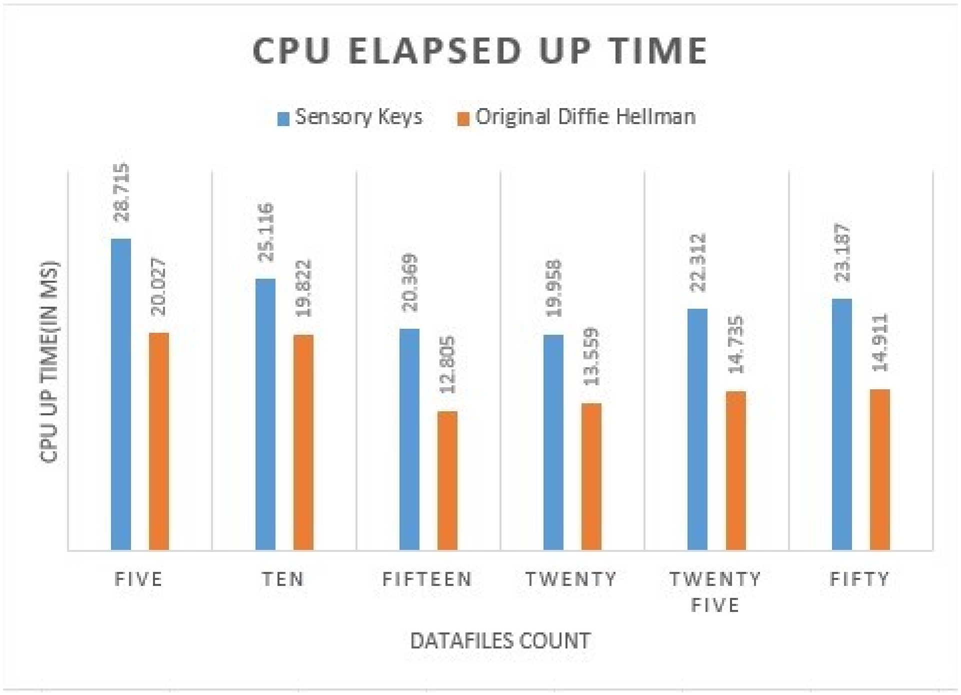


Figure 80 CPU up-time (in milliseconds)

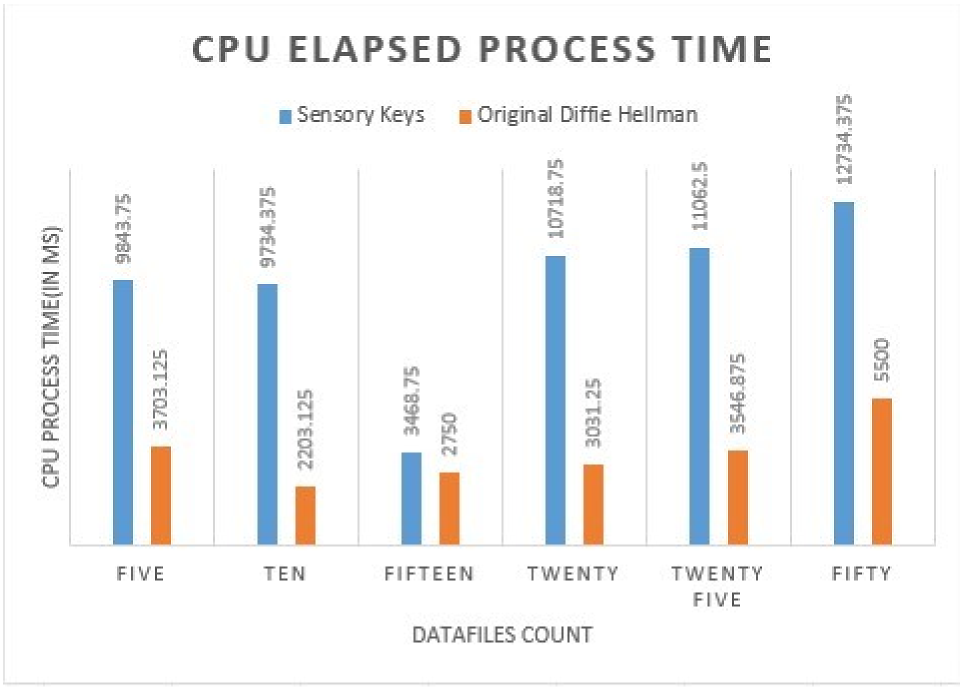


Figure 81 CPU process time (in millisec)

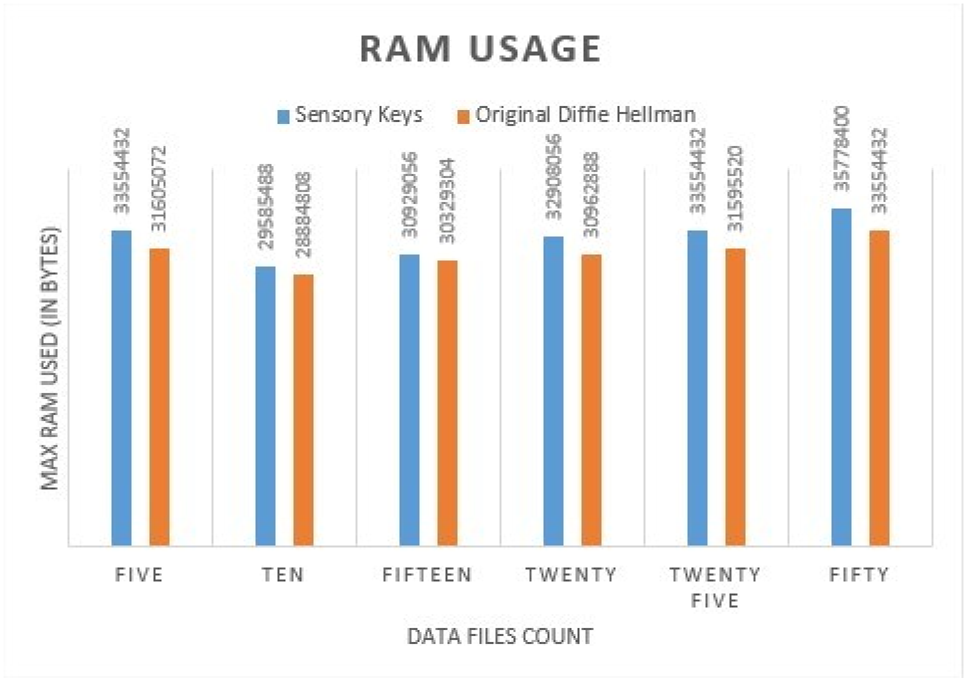


Figure 82 RAM usage (in bytes)

We observe an increase in the RAM usage, CPU up time and the CPU process time due to the involvement of the extra acknowledgements transmission, the searching for the values in the reference frames and the additional hashing operations performed in the proposed approach. The increase observed here is a factor of the complexities involved in the hashing and searching operations which are missing in the implementation of the traditional Diffie Hellman approach. The extra hashing and searching operations are introduced in the proposed approach in order to obtain an improved security without having to use the public key infrastructure. The generation of the Sa, Sb, Ta, Tb and Key remain the same in both the implemented programs.

As in the works of Venkatasubramanian et.al[67] and Sampangi et.al[56], we have also analyzed the *distinctiveness*, *time variance* and *randomness* of the keys generated by the proposed approach.

While analyzing for *Distinctiveness* we looked into the keys to check how unique they are from each other. We have observed that there are identical keys but the identical nature is mainly due to two factors. First is the limitation we encountered during implementing the proposed approach in java wherein we could not store the entire key length into the BigInteger slot and had to work around with only two digit keys. Thus if the actual keys were to be 278 & 279, our implemented program would be using 27 as its keys. Thus, although 278 & 279 are distinct, yet due to the limitation, the end result '27' is not distinct. The second factor influencing the Distinctiveness is, at some instant of time there is a possibility of the person to repeat the same action (such as walking, running etc.) which would yield identical sensor readings and thus identical keys. The likelihood of the second factor is minimal as the speed at which the user walks or runs cannot be identical all the time. Either ways, if we consider a set of keys, they are distinct from one another, i.e. the first key may or may not be the same as that of the second key. In this way, the distinctiveness aspect of the keys is satisfied in the implemented approach.

Time variance indicate that at different instances of time, different keys should be generated. In the proposed approach, the sensors that are used (pressure, gravity, gps, orientation and gyroscope sensors) vary every second. Since there are different sensor values being stored at different instants of time, the keys generated using those values would also tend to be different, preserving the time variant aspect as well.

Randomness implies to the unpredictable and unforeseeable aspect of the keys being used and/or generated. In the proposed algorithm, the randomness of the keys relies on the randomness of the content of the field chosen and also on the randomness of the frame-field chosen. The choice is a function of the outcome of the PRNG which in turn is dependent on the seed used. In the proposed approach, the initial seed is the XOR outcome of the hashes of fingerprint of the user and PUF of the device. The consecutive seeds are the outcome of the PRNG with its seed that was used for the previous iteration. As the result of the PRNG being unpredictable, the frame and field index is unpredictable, thus resulting in the predictability of the value chosen for key generation. In this way, the proposed technique preserves the Randomness nature of the keys.

6.6 SECURITY ANALYSIS

The proposed approach attains Confidentiality and Integrity with the help of encryption. Since the keys are known only to the communicating parties and the adversary is not in possession of the frames and other entities required for the generation of the key, hence snooping into the contents of the data packets would be unfruitful. Added to that, since decryption is technically not possible without the proper set of keys, altering the contents of the transmitted packet is also not possible as each and every packet is encrypted before transmission.

The ‘SAt’ and ‘c1’, helps the server to authenticate the sender (when the server is the receiver) and vice versa. Since the content of the ‘SAt’ and ‘c1’ can be obtained by only the legitimate parties, these entities act as signatures for the message containing them. In this way, Authentication is achieved by the proposed algorithm as only the legitimate parties can be in possession of the values and Non-Repudiation as well.

Access control and Availability are obtained by the use of the fingerprint at the client side which checks for a match of the 'z' value stored after the Registration Phase. It also acts as an authenticating factor at the client side where only the legitimate owner of the device will be authenticated and shall be given access to the application and/or device.

The Scyther evaluation, in Section 6.4, shows that attacks such as interception (i.e. eavesdropping to collect information), modification (i.e. tamper/insert/modify the contents of the data packet), fabrication (i.e. insertion of counterfeit information) are not possible on the data packets during communication.

6.7 SUMMARY OF THE EXPERIMENTAL RESULTS

The results from the Wireshark analysis of the captured traffic shows that, it is difficult for a hacker to backtrack the process or predict the keys. Only the 'Ta' value is transmitted whose contribution is incomplete without the presence of 'p', 'g' and their respective 'Sa' and 'Sb' values towards the Secret Key generation. Thus interception of the traffic is rendered futile for the hacker.

From the test cases considered, the graphs related to the execution time for the experiments discussed in Section 6.3.1 to 6.3.4, show that the execution time increased with the increase in DataFiles count but remained almost same when the DataFiles count was fixed and, the Reference Frames count was increased. Also the graphs displaying the key values used for encrypting each DataFiles for all the test cases discussed in Section 6.3, showed that the keys generated are random and un-predictable. None of the graph's pattern is duplicated in any other graph for any of the test cases.

Scyther validates and states that our proposed approach is free from all major attacks. All the parameters which are transferred over the communication channel are secure and safe from the grasp of a hacker.

The performance analysis shows us that the proposed algorithm consumes relatively greater system resources when compared to the customary Diffie Hellman due to the inclusion of the addition set of steps to secure the algorithm. Also the keys generated are distinct, time variant and random. The security analysis shows that confidentiality, integrity, authentication, access control and availability are satisfied with the help of the entities generated and used in the proposed approach. Figure 83 explains how the proposed approach helps in reducing the possibility of MITM in Diffie Hellman approach. The values represented in blue color are not known to the adversary and the values in red are known to the adversary. Since the adversary is not in possession of p, g, S_a, S_b of the parties exchanging the data, hence he cannot generate the Key required to decrypt the data. Also each acknowledgement is attached with an authentication factor which would authenticate the sender. Hence any attempt to send false acknowledgment or false values would be identified and the respective packet will be dropped.

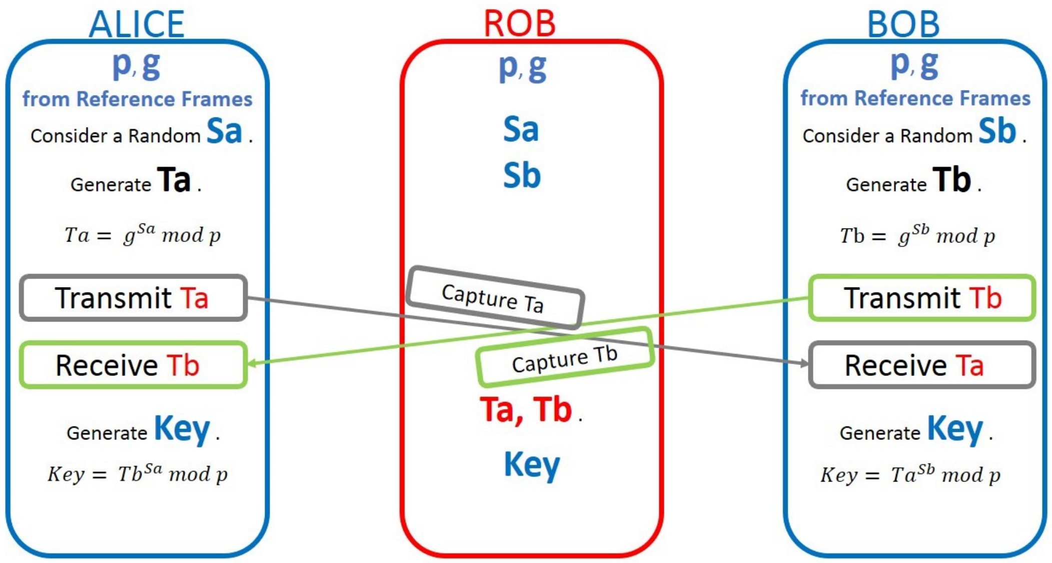


Figure 83 Prevention of MITM using the Proposed Approach

CHAPTER 7 CONCLUSION

As per the actual description of the Diffie Hellman algorithm, the entire technique for the Secret Key generation leaves it susceptible to MITM attacks. Several approaches have been proposed by various authors with improvements to the original approach in order to defend the attack. The main focus lays in randomization of the values used to calculate the Keys by both the parties.

This research work, primarily focuses on reducing the probability of MITM attack using a modified Diffie Hellman approach. Lee et.al [71], [72], Stulman et.al [68], Shen et.al [69] and Yang et.al [70] throw light on the possibility of the occurrences of MITM. [71] & [72] specifically discuss how MITM can be launched in the Smartphone world by exploiting the vulnerabilities such as the incapability of the devices to appropriately verify the Certificates of the CA's. Also the CA's are being compromised as well making the reliability on them even more harder [68]. In [72], the author speak about how the users can be tricked to install applications which would leave the device vulnerable to MITM attacks. The underlying procedure for launching the attack remains the same as discussed in Section 2.5.1 but the technique followed to implant the bug which would render the verification of the certificates useless, changes from one adversary to another. Thus this calls for an approach which would require no prior knowledge of the communicating parties along with the need to eliminate the transfer of the key over the communication channel. Hence the proposed approach includes the Diffie Hellman approach for communication. Added to that, we also intended to secure the data in case of brute force attempts. Towards that goal, we strived to keep the key as random and as unpredictable as possible. In addition, the key updates itself for each cycle of data being transferred.

The proposed approach makes use of the sensor data that is accumulated using the sensors available in the devices. Since it is difficult to predict the behavior of the user, the prediction of the values being generated by the sensor hardware becomes equally difficult. The proposed approach does not necessarily need a sensor value. The sensors can be substituted with any hardware whose value fluctuates frequently. For instance, in devices such as Access points, the Received Signal Strength Indicator (RSSI) values, frequency etc. values can be used.

The collected values are stored in the Reference Frames in an array format. The contents of the array are used as the values for the various terms required for the Key generation and for other purposes such as generation of Acknowledgments etc. ‘p’ and ‘g’ are not transferred over the communication channel. Only the ‘Ta’ value and the ‘Tb value are exchanged. Both the parties generate the required terms at their respective ends.

The Proposed approach involves two phases; a Registration Phase, which occurs only once in the entire lifetime of the application and a Data Transfer Phase, which is responsible for the key generation and encryption of the data to be transferred. The Registration Phase plays its role at the time of installation during when the set of values required for the Key generation and other security aspects facilitated by the approach, are transferred and stored. Once the data is stored, for each of the data transferred, these are updated to prevent the hacker from gaining a pattern of the keys being used.

The user’s fingerprint and the device’s unique characteristic ‘PUF’ are used together to provide access control. After the user provides his fingerprint, if a match is not found then access to the application is not provided to the accessor. Furthermore, two way acknowledgments are used to ensure integrity in both the phases. Forward and Backward secrecy are also achieved by the end of the acknowledgment transfer, as all the keys are deleted after the reception of the acknowledgment.

The researcher performed a set of test cases which mimic the activities of a user in real time scenario and the data is collected. Then the proposed approach is tested and the results are analyzed. The execution times for the same DataFiles count and different Reference Frames remain pretty much the same. This implies that the Reference Frame count does not influence the time taken for completion of the transfer of Data but with the increase in Reference Frames count, it does increase the choices from which the values are chosen. Moreover, each of the key graph have a unique pattern i.e. no graph’s pattern is similar to any other graph's pattern. Of the keys which are repeating in the test cases, predicting when the keys would repeat itself is difficult. The reason the keys are repeating is due to the implementation limitation we encountered. If the whole of the content of Reference Frames are used, then the randomness can also be increased.

From the traffic analyzed using Wireshark, no element of the message being transferred, reveal the key nor do they support in key generation directly. In short it would be of no benefit to the hacker even if he intercepts the ongoing communication.

Scyther results demonstrate that there are no possible attacks which could compromise the proposed algorithm. Neither the Registration Phase nor the Data Transfer Phase's contents were attacked or leaked as per the tests performed by the Scyther tool. In essence, Scyther claims the proposed algorithm to be secure.

The increased number of steps and instructions in the proposed approach when compared to that of the traditional Diffie Hellman approach has forced the algorithm to consume relatively more system resources. But in return we are facilitated with attainment of various security goals such as confidentiality, authentication, integrity, access control and availability making the proposed approach more secure.

Summing up the thesis work, the proposed approach greatly reduces the probability the occurrence of MITM attack using a modified Diffie Hellman algorithm. The proposed algorithm also facilitates Access Control and Forward & Backward Secrecy. Prediction of Keys is difficult and two-way acknowledgements ensure a successful data transfer as well. Transmission of lesser values needed for Key generation renders interception of the traffic futile. Sequence numbers help prevent Replay attacks and the possession of the Reference Frames by the end parties only, provide non-repudiation, as the contents are not open to the public.

7.1 Limitations

The proposed approach was tested in Scyther which is a Formal Protocol analyzer. It does not check for synchronization attack or replay attacks. Thus this thesis work does not check for the possibility successfully defending such attacks. Although we have included sequence number to prevent replay attacks yet we have no solid evidence to prove that our approach is resistant to such attack. This is not exactly a limitation of the approach but a limitation encountered due to the limitation of the Scyther tool itself. In addition, the current implementation limits the key size to maximum of 2 digits due to the limitation encountered during the implementation of the algorithm. This, yet again is a limitation of the BigInteger data structure than the limitation of the approach itself.

7.2 Discussion and Future work

In the proposed algorithm, 'S_a' and 'S_b' are generated randomly. This can be substituted with one of the reference values using the same technique proposed in this thesis. This should eliminate the need to transmit 'T_a' and 'T_b' over the wire. The 'T_a' and 'T_b' can be generated by each of the parties independently and thus the keys. Only the authentic parties would be in possession of the keys using which the message can be encrypted or decrypted. This avoidance of exchanging the 'T_a' and 'T_b' values would also reduce the overhead and the extra cost thus incurred. Without the presence of 'T_a' and 'T_b' values, it would be even more difficult for the attacker to predict the keys or launch MITM attack. The entire algorithm with the mentioned changes can be implemented and tested again to check for the possibility of any attacks.

As far as the proposed approach is concerned, we would like to design a scenario where the Replay attack can be launched and observed in order to check if the proposed approach is defending it or not. We would also like to make use of the entire content of the sensor value for the generation of the key i.e. keys of length greater than two and then test for the presence of randomness in the generated keys. Based on our observation, we have not noticed any drastic hike in the execution time as the number of Reference Frames increased until 25. Further testing is required in order to observe or comment on the execution time consumption for Reference Frame counts beyond 25. Also the key sizes are restricted to two digits due to the implementation limitation we encountered. We would also like to analyze the security aspect of different algorithms with various recommended key lengths such as 256 bits, 512 bits, 1024 bits and so on.

REFERENCES

[1] Statista, "Statista.com," [Online]. Accessed June 6, 2015 from http://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/.

[2] Salesforce.Inc, "Salesforce App Mobile Cloud," Salesforce, [Online]. Accessed February 23, 2016 from https://developer.salesforce.com/mobile.

[3] Dropbox, "Dropbox Help Center," [Online]. Accessed March 19, 2016 from https://www.dropbox.com/en/help/27.

[4] Hirani, S. A. (2003). Energy consumption of encryption schemes in wireless devices (Doctoral dissertation, University of Pittsburgh).

[5] Diffie, W., & Hellman, M. E. (1976, June). Multiuser cryptographic techniques. In Proceedings of the June 7-10, 1976, national computer conference and exposition (pp. 109-112). ACM.

[6] Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). Handbook of applied cryptography. CRC press.

[7] Milanov, E. (2009). The rsa algorithm. RSA Laboratories.

[8] Maes, R. (2013). Physically Unclonable Functions. Springer, Berlin.

[9] R. Goodrich, "liveScience," [Online]. Accessed April 09, 2015 from http://www.livescience.com/40102-accelerometers.html.

[10] N. Chandler, "HowStuffWorks," [Online]. Accessed July 25, 2015 from http://electronics.howstuffworks.com/gadgets/fitness/fitbit.htm.

[11] Suh, G. E., & Devadas, S. (2007, June). Physical unclonable functions for device authentication and secret key generation. In Proceedings of the 44th annual Design Automation Conference (pp. 9-14). ACM.

[12] Ayushi, "A Symmetric Key Cryptographoc Algorithm," in International Journal of Computer Applications, 2010.

[13] Merkle, R. C. (1978). Secure communications over insecure channels. Communications of the ACM, 21(4), 294-299.

[14] Diffie, W., & Hellman, M. (1976). New directions in cryptography. IEEE transactions on Information Theory, 22(6), 644-654.

[15] Wikipedia, "Diffie Hellman Key Exchange," [Online]. Accessed August 15, 2015 from https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange.

[16] Shafiq, M. Z., Ji, L., Liu, A. X., Pang, J., & Wang, J. (2012). A first look at cellular machine-to-machine traffic: large scale measurement and characterization. ACM SIGMETRICS Performance Evaluation Review, 40(1), 65-76.

[17] Maier, G., Schneider, F., & Feldmann, A. (2010, April). A first look at mobile hand-held device traffic. In International Conference on Passive and Active Network Measurement (pp. 161-170). Springer Berlin Heidelberg.

[18] Li, Y., Yang, J., & Ansari, N. (2014, June). Cellular smartphone traffic and user behavior analysis. In 2014 IEEE International Conference on Communications (ICC) (pp. 1326-1331). IEEE.

[19] Liu, J., Wu, H., & Wang, H. (2014, September). A detection method for malicious codes in Android apps. In Wireless Communications, Networking and Mobile Computing (WiCOM 2014), 10th International Conference on (pp. 514-519). IET.

[20] Louk, M., Lim, H., & Lee, H. (2014). An analysis of security system for intrusion in smartphone environment. The Scientific World Journal, 2014.

[21] Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011, October). Crowdroid: behavior-based malware detection system for android. In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices (pp. 15-26). ACM.

[22] Finickel, E., Lahmadi, A., Beck, F., & Festor, O. (2014, June). Empirical analysis of Android logs using self-organizing maps. In 2014 IEEE International Conference on Communications (ICC) (pp. 1802-1807). IEEE.

[23] Hirabe, Y., Arakawa, Y., & Yasumoto, K. (2014, January). Logging all the touch operations on Android. In Mobile Computing and Ubiquitous Networking (ICMU), 2014 Seventh International Conference on (pp. 93-94). IEEE.

[24] Tian, J., Wang, G., Gao, X., & Shi, K. (2014, May). User behavior based automatical navigation system on android platform. In 2014 23rd Wireless and Optical Communication Conference (WOCC) (pp. 1-6). IEEE.

[25] Datta, S. K., Bonnet, C., & Nikaein, N. (2012, June). Android power management: Current and future trends. In Enabling Technologies for Smartphone and Internet of Things (ETSIoT), 2012 First IEEE Workshop on (pp. 48-53). IEEE.

[26] Ma, Y., Xu, B., Bai, Y., Sun, G., & Zhu, R. (2012, May). Daily mood assessment based on mobile phone sensing. In 2012 ninth international conference on wearable and implantable body sensor networks (pp. 142-147). IEEE.

[27] Bedogni, L., Di Felice, M., & Bononi, L. (2012, November). By train or by car? Detecting the user's motion type through smartphone sensors data. In Wireless Days (WD), 2012 IFIP (pp. 1-6). IEEE.

[28] Bo, C., Zhang, L., Jung, T., Han, J., Li, X. Y., & Wang, Y. (2014, December). Continuous user identification via touch and movement behavioral biometrics. In 2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC) (pp. 1-8). IEEE.

[29] Islam, M., Shah, M., Khan, Z., Mahmood, T., & Khan, M. J. (2015, December). A New Symmetric Key Encryption Algorithm using Images as Secret Keys. In 2015 13th International Conference on Frontiers of Information Technology (FIT) (pp. 1-5). IEEE.

[30] Khader, A. S., & Lai, D. (2015, April). Preventing man-in-the-middle attack in Diffie-Hellman key exchange protocol. In Telecommunications (ICT), 2015 22nd International Conference on (pp. 204-208). IEEE.

[31] Ahmad, S., Alam, K. M. R., Rahman, H., & Tamura, S. (2015, January). A comparison between symmetric and asymmetric key encryption algorithm based decryption mixnets. In Networking Systems and Security (NSysS), 2015 International Conference on (pp. 1-5). IEEE.

[32] Wang, C., Han, Y., & Li, F. (2010, April). New Signcryption from q-Diffie-Hellman Problems. In Communications and Mobile Computing (CMC), 2010 International Conference on (Vol. 1, pp. 35-40). IEEE.

[33] Sha, K., Xu, C., & Wang, Z. (2014, August). One-time symmetric key based cloud supported secure smart meter reading. In 2014 23rd International Conference on Computer Communication and Networks (ICCCN) (pp. 1-6). IEEE.

[34] Khan, E., Gabidulin, E., Honary, B., & Ahmed, H. (2012). Matrix-based memory efficient symmetric key generation and pre-distribution scheme for wireless sensor networks. IET wireless sensor systems, 2(2), 108-114.

[35] Fuloria, S., Anderson, R., Alvarez, F., & McGrath, K. (2011, March). Key management for substations: Symmetric keys, public keys or no keys?. In Power Systems Conference and Exposition (PSCE), 2011 IEEE/PES (pp. 1-6). IEEE.

[36] Kumar, C. K., Jose, G. J. A., Sajeev, C., & Suyambulingom, C. (2012). Safety measures against man-in-the-middle attack in key exchange. ARPN Journal of Engineering and Applied Sciences, 7(2).

[37] Van Oorschot, P. C., & Wiener, M. J. (1996, May). On Diffie-Hellman key agreement with short exponents. In International Conference on the Theory and Applications of Cryptographic Techniques (pp. 332-343). Springer Berlin Heidelberg.

[38] Chang, R. Y., Lin, S. J., & Chung, W. H. (2013, October). Diffie-Hellman key distribution in wireless multi-way relay networks. In Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific (pp. 1-4). IEEE.

[39] Quisquater, J. J., Quisquater, M., Quisquater, M., Quisquater, M., Guillou, L., Guillou, M. A., ... & Guillou, S. (1989, August). How to explain zero-knowledge protocols to your children. In Conference on the Theory and Application of Cryptology (pp. 628-631). Springer New York.

[40] Liang, B., & Wu, Z. (2014, November). A Novel Fingerprint-Based Biometric Encryption. In P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2014 Ninth International Conference on (pp. 146-150). IEEE.

[41] Jain, A. K., & Uludag, U. (2003, July). Multimedia content protection via biometric based encryption. In Proc. of IEEE International Conference on Multimedia and Expo, ICME.

[42] Sharma, S., & Balasubramanian, V. (2014, November). A biometric based authentication and encryption Framework for Sensor Health Data in Cloud. In Information Technology and Multimedia (ICIMU), 2014 International Conference on (pp. 49-54). IEEE.

[43] Wang, Z., Dou, R., Leng, Y., & Wang, J. (2010, July). A new framework of Biometric encryption with filter-bank based fingerprint feature. In Signal Processing Systems (ICSPS), 2010 2nd International Conference on (Vol. 3, pp. V3-169). IEEE.

[44] Mejri, M. N., Achir, N., & Hamdi, M. (2016, January). A new group Diffie-Hellman key generation proposal for secure VANET communications. In 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC) (pp. 992-995). IEEE.

[45] Zhang, X., Ma, S., Han, D., & Shi, W. (2015, January). Implementation of elliptic curve Diffie-Hellman key agreement scheme on IRIS nodes. In Intelligent Computing and Internet of Things (ICIT), 2014 International Conference on (pp. 160-163). IEEE.

[46] Wikipedia, "Wikipedia," [Online]. Accessed April 09, 2016 from https://en.wikipedia.org/wiki/Java_(programming_language).

[47] Oracle, "Oracle- Java Security," [Online]. Accessed November 06, 2015 from http://docs.oracle.com/javase/7/docs/technotes/guides/security/overview/jsoverview.html.

[48] Oracle, "Oracle-java.net," [Online]. Accessed November 06, 2015 from https://docs.oracle.com/javase/7/docs/api/java/net/package-summary.html.

[49] TutorialsPoint, "TutorialsPoint- Collection Framework," [Online]. Accessed October 02, 2015 from http://www.tutorialspoint.com/java/java_collections.htm.

[50] Oracle, "Oracle- Class Secure Random," [Online]. Accessed November 06, 2016 from https://docs.oracle.com/javase/7/docs/api/java/security/SecureRandom.html.

[51] GSMArena, "GSM Arena," [Online]. Accessed June 06, 2015 from http://www.gsmarena.com/samsung_galaxy_s6-6849.php.

[52] RawCap, "Netresec," [Online]. Accessed January 15, 2016 from http://www.netresec.com/?page=RawCap.

[53] Wireshark, "Wireshark," [Online]. Accessed January 15, 2016 from https://www.wireshark.org/.

[54] C. Cremers, Scyther User Manual, Department of Computer Science, University of Oxford. [Online]. Available: https://github.com/cascremers/ scyther/blob/master/gui/scyther-manual.pdf

[55] "An Overview of the Android Architecture". Accessed December 25, 2016, from http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture

[56] Sampangi, R. V., Dey, S., Urs, S. R., & Sampalli, S. (2012). A security suite for wireless body area networks. arXiv preprint arXiv:1202.2171.

[57] Pawar, M. V., & Anuradha, J. (2015). Network Security and Types of Attacks in Network. Procedia Computer Science, 48, 503-506.

[58] Anwar, R. W., Bakhtiari, M., Zainal, A., Abdullah, A. H., & Qureshi, K. N. (2014). Security issues and attacks in wireless sensor network. World Applied Sciences Journal, 30(10), 1224-1227.

[59] "Android Developer," Google, [Online]. Accessed December 25, 2016 from https://developer.android.com/about/versions/marshmallow/android-6.0.html.

[60]"Dreamstime.com," [Online]. Accessed April 09, 2016 https://www.dreamstime.com/stock-illustration-phone-evolution-telephone-communication-progress-mobile-classic-device-vector-illustration-image53386436.

[61]"Yaldex," [Online]. Accessed March 05, 2016 from http://www.yaldex.com/tcp_ip/0672325659_ch20lev1sec1.html.

[62] C. Cremers. The Scyther tool. Department of Computer Science, University of Oxford. [Online]. Available: http://users.ox.ac.uk/~coml0529/scyther/

[63] Mitrokotsa, A., Rieback, M. R., & Tanenbaum, A. S. (2010). Classifying RFID attacks and defenses. Information Systems Frontiers, 12(5), 491-505.

[64] Chaudhry, J., Qidwai, U. A., Rittenhouse, R. G., & Lee, M. (2012, October). Vulnerabilities and verification of cryptographic protocols and their future in wireless body area networks. In Emerging Technologies (ICET), 2012 International Conference on (pp. 1-5). IEEE.

[64] P. Chatterjee, "DigiKey," Yolo Development, [Online]. Accessed May 05, 2016 from http://www.digikey.com/en/articles/techzone/2011/sep/constructing-mobile-multi-sensor-systems-dominated-by-a-touch-display.

[65] "Android", Operating System,[Online]. Accessed June 10, 2016 from https://en.wikipedia.org/wiki/Android_(operating_system)

[66] A. Smith, U.S. Smartphone use in 2015, "PEW Research Center" [Online], Accessed June 06, 2016 from http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/pi_2015-04-01_smartphones_03/

[67] K. Venkatasubramanian, Venkatasubramanian, A. Banerjee, and S. Gupta,"EKG-based key agreement in body sensor networks," in IEEE INFOCOM Workshops 2008, April 2008, pp. 1-6.

[68] Stulman, A., Lahav, J., & Shmueli, A. (2012, December). Manet secure key exchange using spraying diffie-hellman algorithm. In Internet Technology And Secured Transactions, 2012 International Conference for (pp. 249-252). IEEE.

[69] Shen, W., Hong, W., Cao, X., Yin, B., Shila, D. M., & Cheng, Y. (2014, December). Secure key establishment for device-to-device communications. In 2014 IEEE Global Communications Conference (pp. 336-340). IEEE.

[70] Yang, G. M., Chen, J. M., Lu, Y. F., & Ma, D. M. (2010, March). An efficient improved group key agreement protocol based on Diffie-Hellman key exchange. In Advanced Computer Control (ICACC), 2010 2nd International Conference on (Vol. 2, pp. 303-306). IEEE.

[71] Lee, J., Tu, C., & Jung, S. (2012). Man-in-the-middle Attacks Detection Scheme on Smartphone using 3G network. In The Fourth International Conference on Evolving Internet (pp. 65-70).

[72] Cambridge, "Akamia", IOS And Android OS Targeted By Man-In-The-Middle Attacks,[Online]. Accessed August 15, 2016 from https://www.akamai.com/us/en/about/news/press/2014-press/ios-and-android-os-targeted-by-man-in-the-middle-attacks.jsp

APPENDIX A

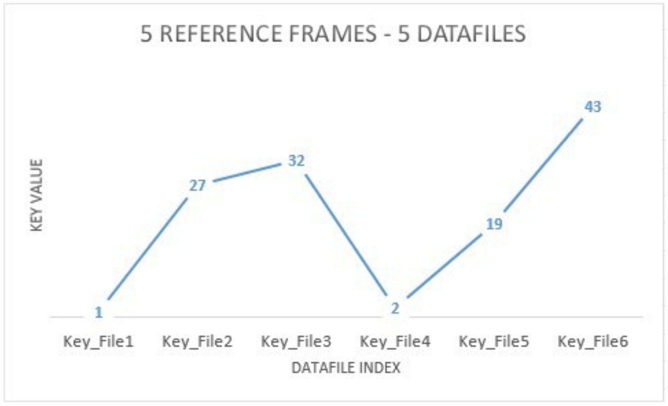


Figure 84 Key value obtained with 5 Reference Frames for 5 DataFiles for Simulated Data

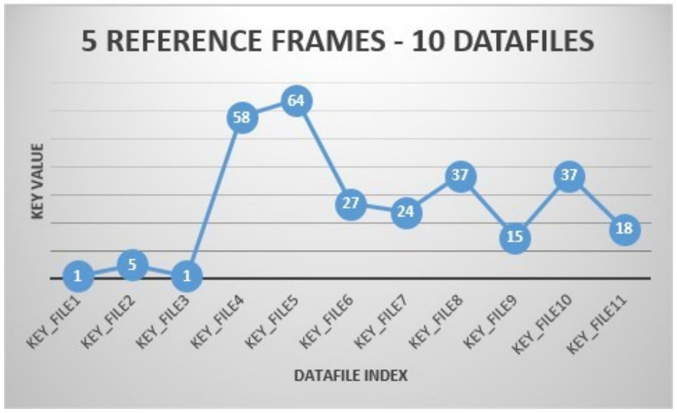


Figure 85 Key value obtained with 5 Reference Frames for 10 DataFiles for Simulated Data

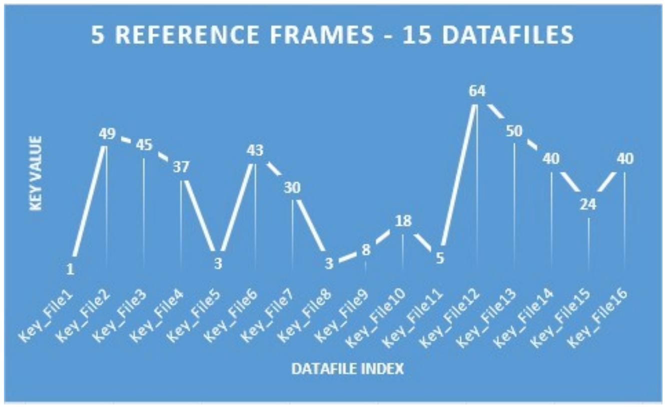


Figure 86 Key value obtained with 5 Reference Frames for 15 DataFiles for Simulated Data

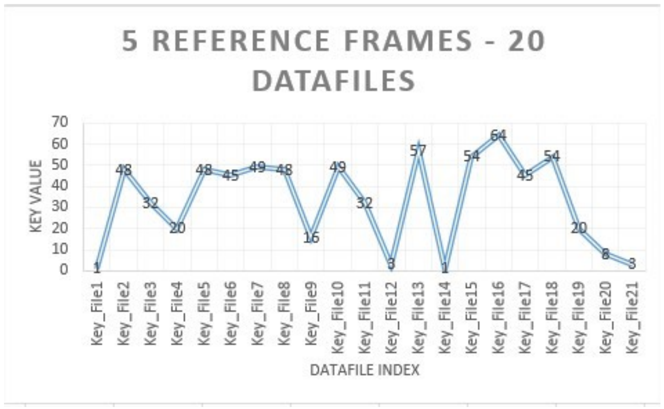


Figure 87 Key value obtained with 5 Reference Frames for 20 DataFiles for Simulated Data

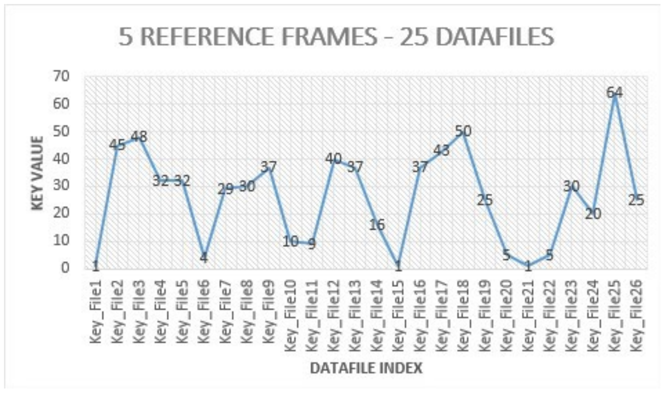


Figure 88 Key value obtained with 5 Reference Frames for 25 DataFiles for Simulated Data

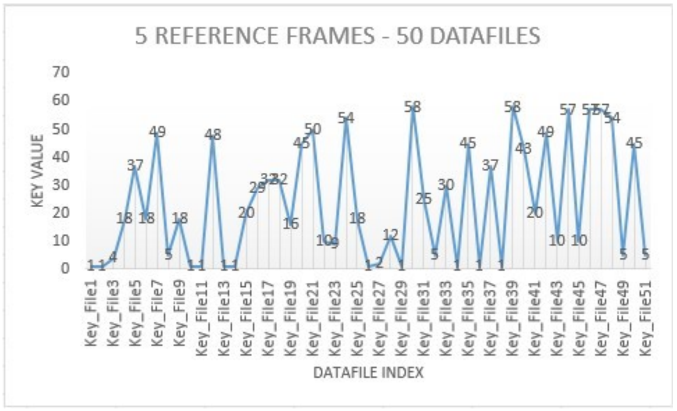


Figure 89 Key value obtained with 5 Reference Frames for 50 DataFiles for Simulated Data

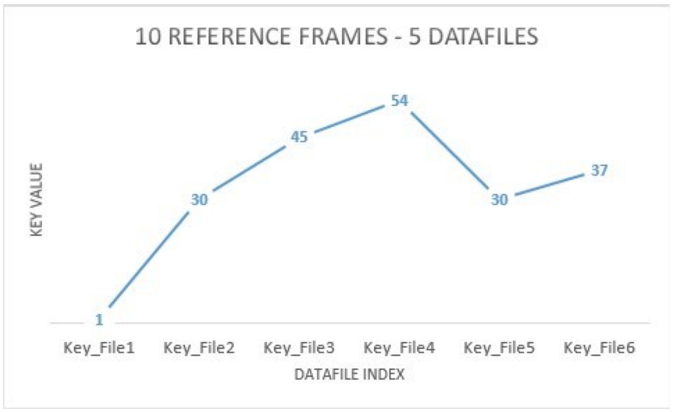


Figure 90 Key value obtained with 10 Reference Frames for 5 DataFiles for Simulated Data

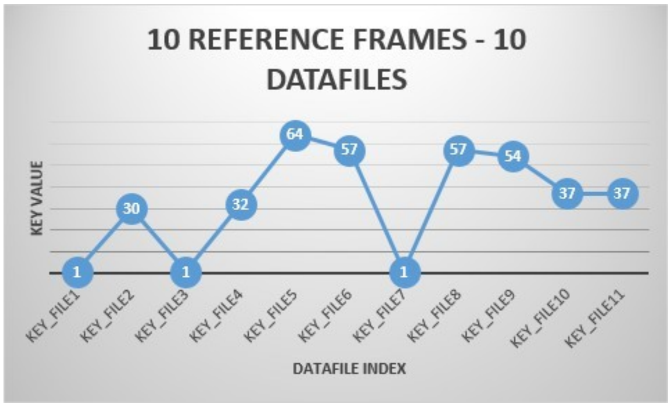


Figure 91 Key value obtained with 10 Reference Frames for 10 DataFiles for Simulated Data

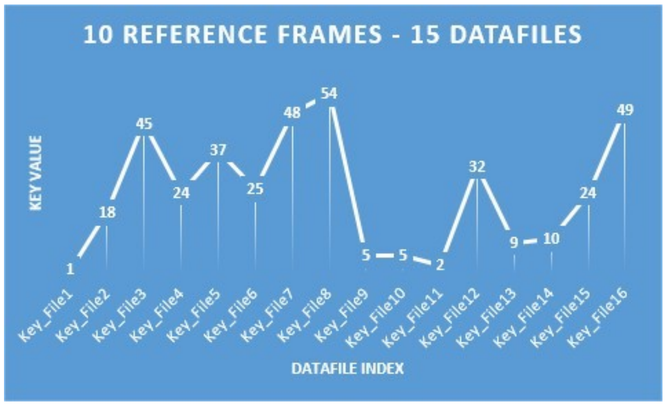


Figure 92 Key value obtained with 10 Reference Frames for 15 DataFiles for Simulated Data

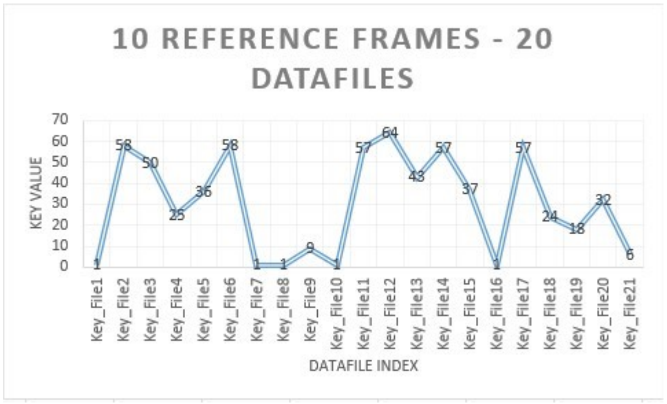


Figure 93 Key value obtained with 10 Reference Frames for 20 DataFiles for Simulated Data

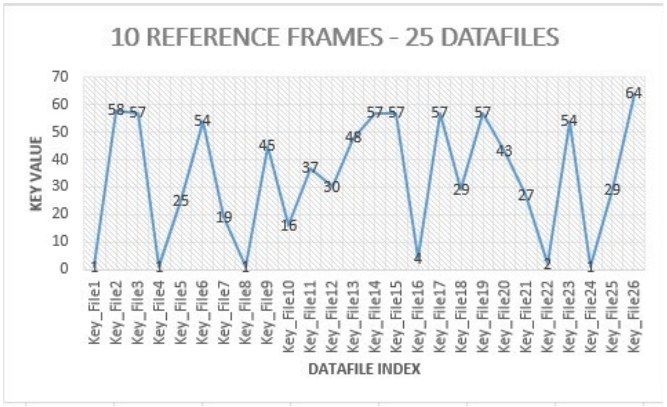


Figure 94 Key value obtained with 10 Reference Frames for 25 DataFiles for Simulated Data

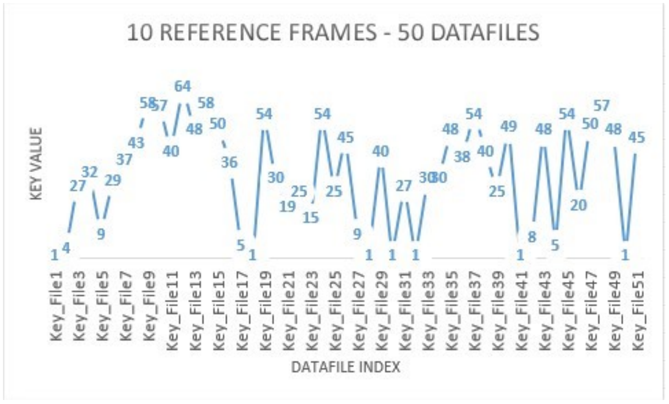


Figure 95 Key value obtained with 10 Reference Frames for 50 DataFiles for Simulated Data

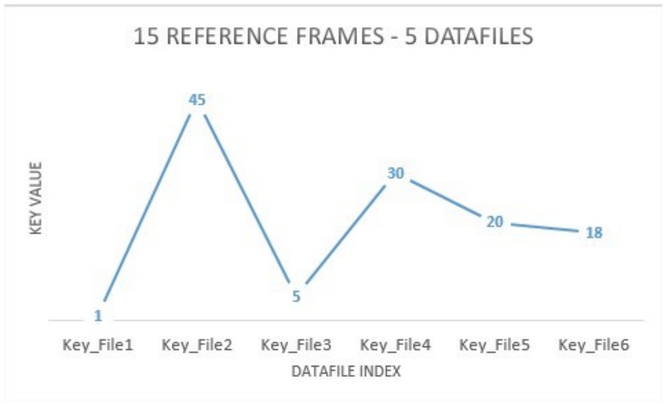


Figure 96 Key value obtained with 15 Reference Frames for 5 DataFiles for Simulated Data

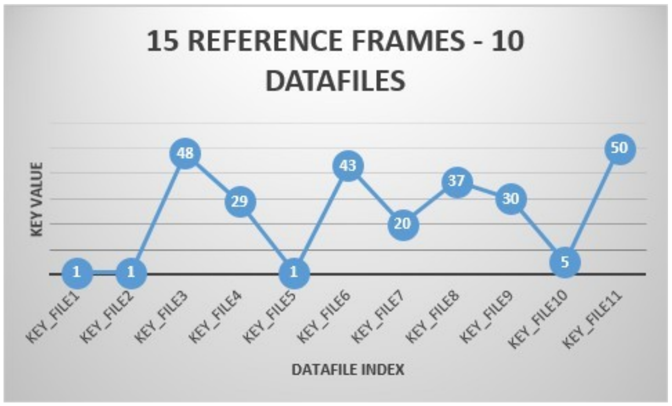


Figure 97 Key value obtained with 15 Reference Frames for 10 DataFiles for Simulated Data

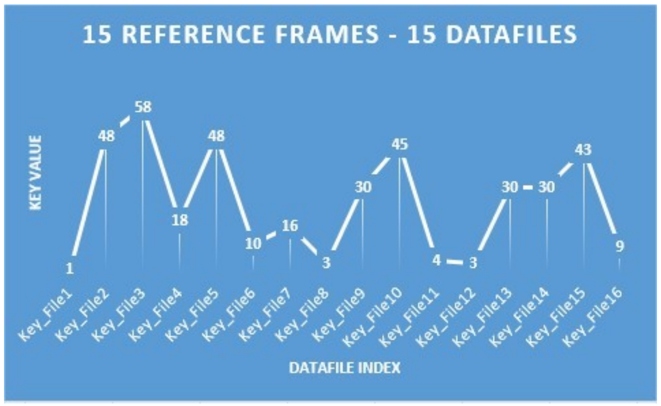


Figure 98 Key value obtained with 15 Reference Frames for 15 DataFiles for Simulated Data

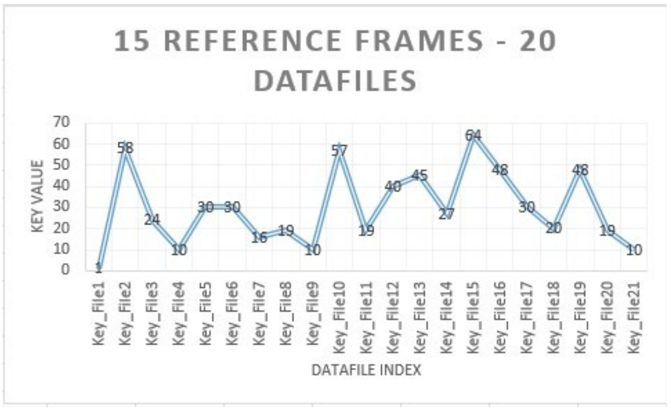


Figure 99 Key value obtained with 15 Reference Frames for 20 DataFiles for Simulated Data

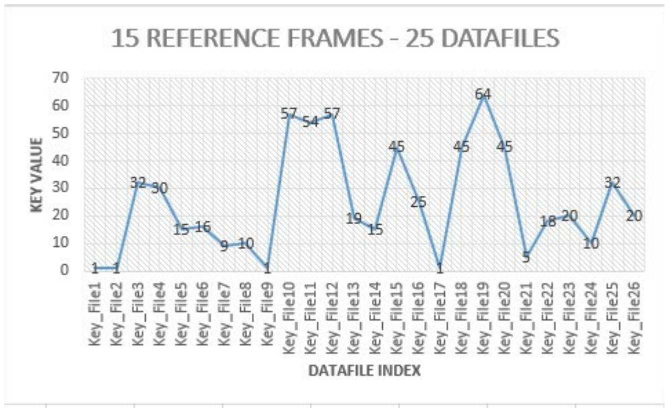


Figure 100 Key value obtained with 15 Reference Frames for 25 DataFiles for Simulated Data

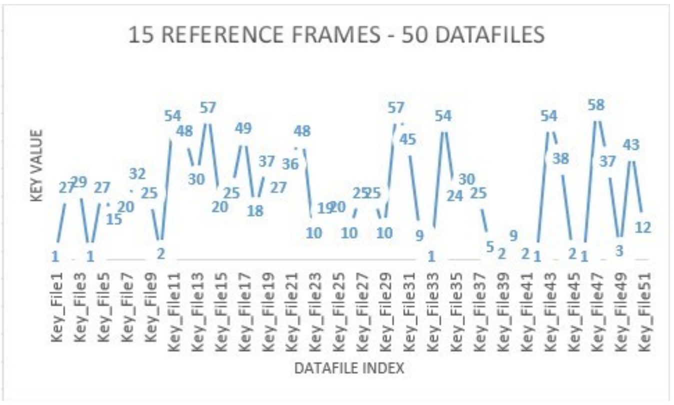


Figure 101 Key value obtained with 15 Reference Frames for 50 DataFiles for Simulated Data

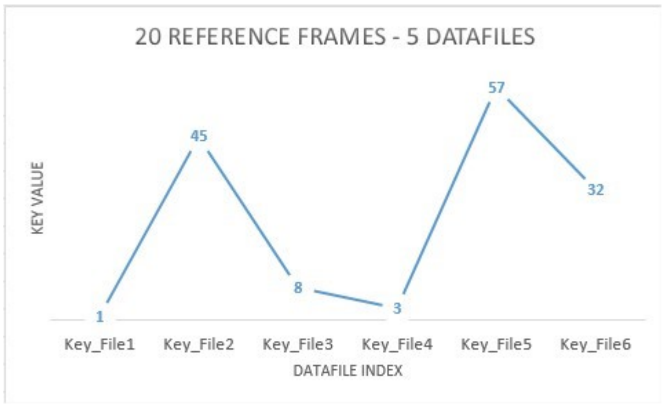


Figure 102 Key value obtained with 20 Reference Frames for 5 DataFiles for Simulated Data

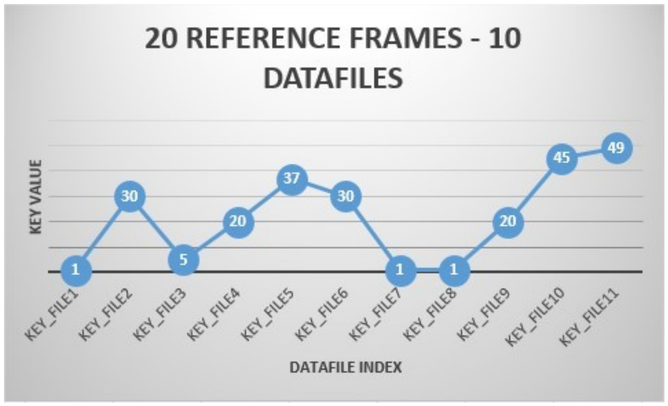


Figure 103 Key value obtained with 20 Reference Frames for 10 DataFiles for Simulated Data

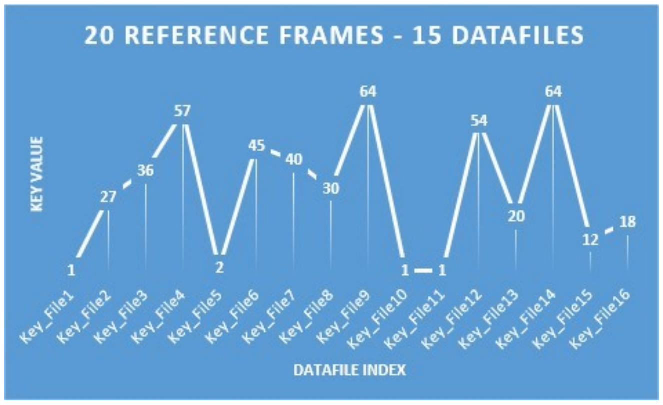


Figure 104 Key value obtained with 20 Reference Frames for 15 DataFiles for Simulated Data

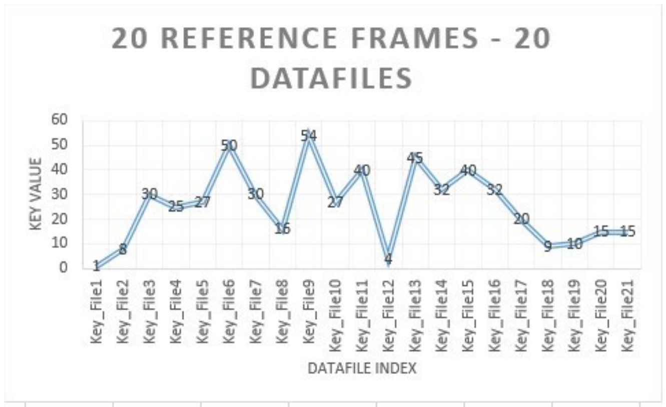


Figure 105 Key value obtained with 20 Reference Frames for 20 DataFiles for Simulated Data

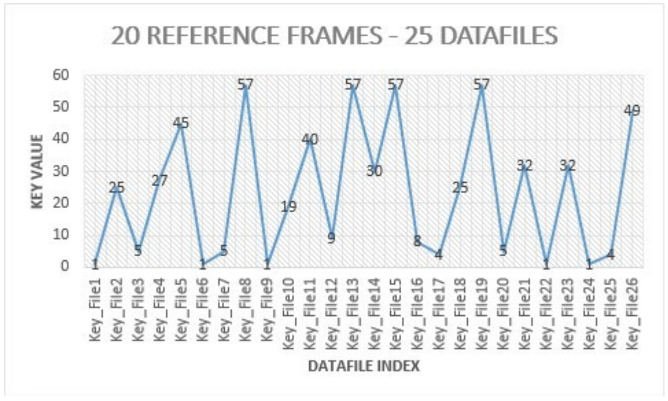


Figure 106 Key value obtained with 20 Reference Frames for 25 DataFiles for Simulated Data

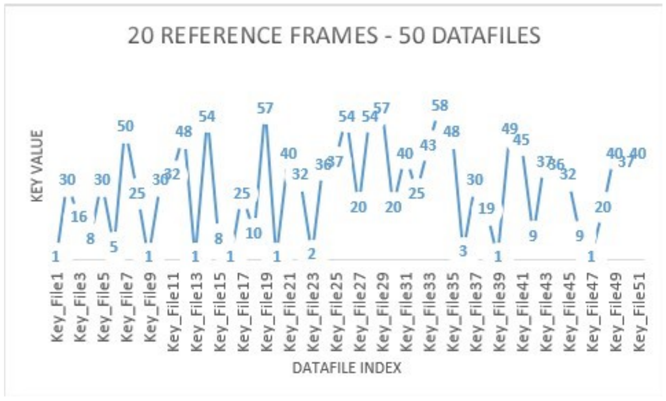


Figure 107 Key value obtained with 20 Reference Frames for 50 DataFiles for Simulated Data

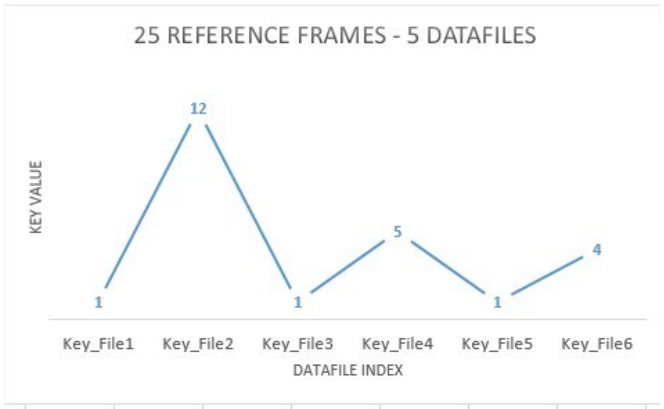


Figure 108 Key value obtained with 25 Reference Frames for 5 DataFiles for Simulated Data

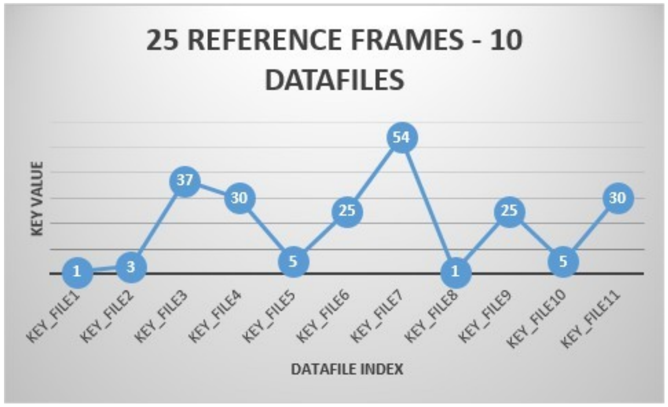


Figure 109 Key value obtained with 25 Reference Frames for 10 DataFiles for Simulated Data

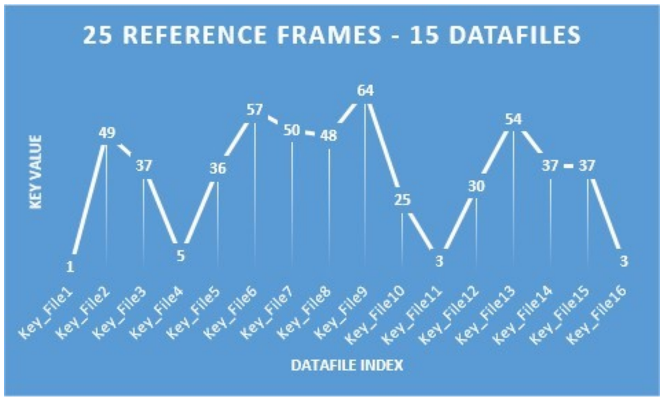


Figure 110 Key value obtained with 25 Reference Frames for 15 DataFiles for Simulated Data

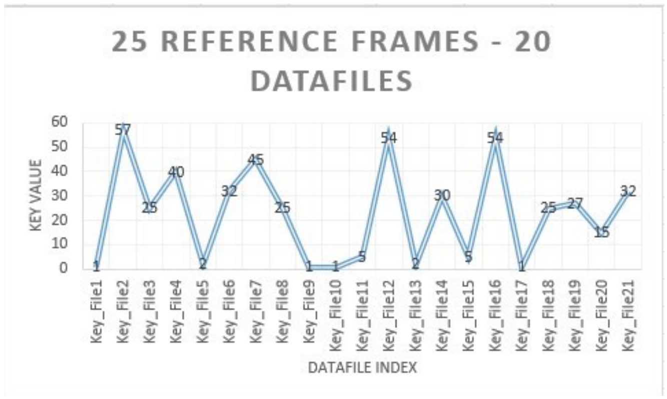


Figure 111 Key value obtained with 25 Reference Frames for 20 DataFiles for Simulated Data

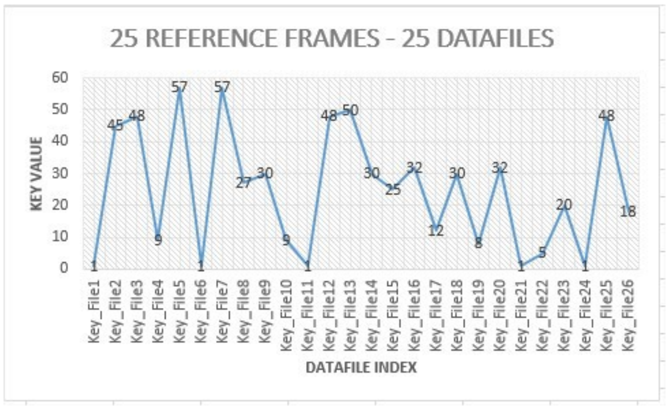


Figure 112 Key value obtained with 25 Reference Frames for 25 DataFiles for Simulated Data

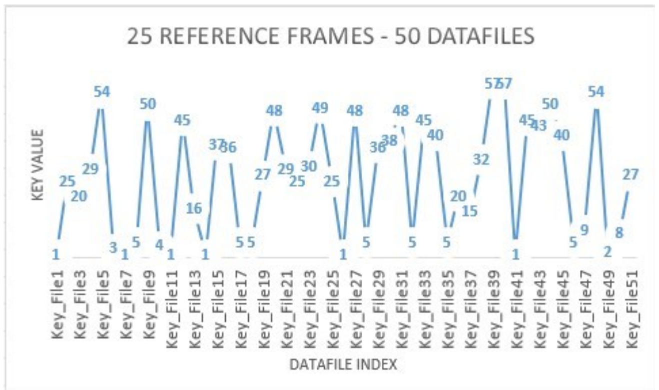


Figure 113 Key value obtained with 25 Reference Frames for 50 DataFiles for Simulated Data

APPENDIX B

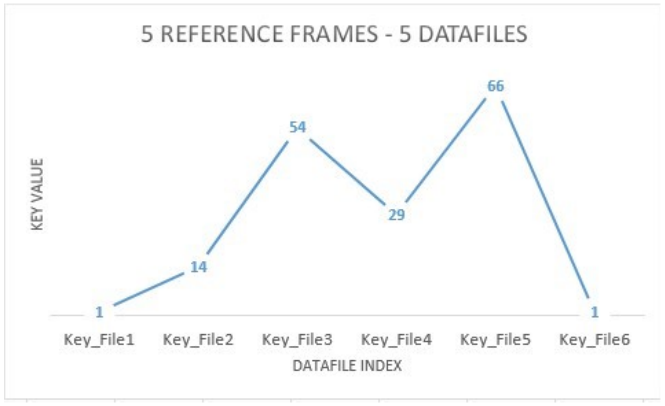


Figure 114 Key value obtained with 5 Reference Frames for 5 DataFiles for Vertical movement of the Device

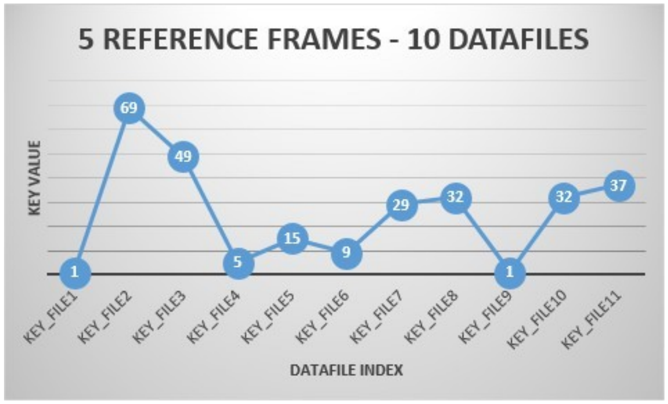


Figure 115 Key value obtained with 5 Reference Frames for 10 DataFiles for Vertical movement of the Device

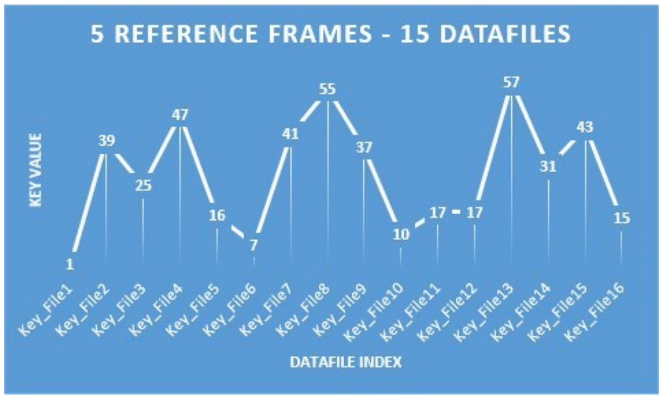


Figure 116 Key value obtained with 5 Reference Frames for 15 DataFiles for Vertical movement of the Device.

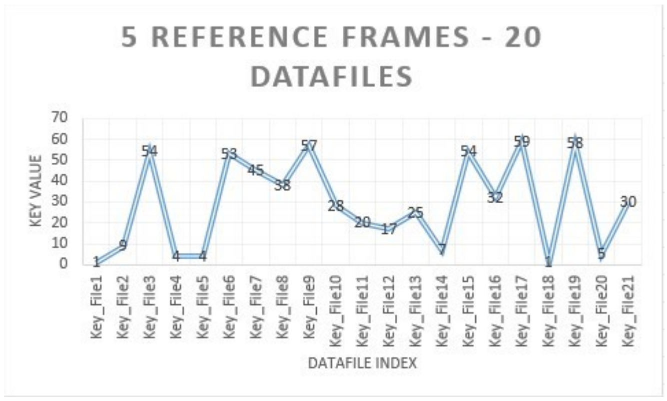


Figure 117 Key value obtained with 5 Reference Frames for 20 DataFiles for Vertical movement of the Device

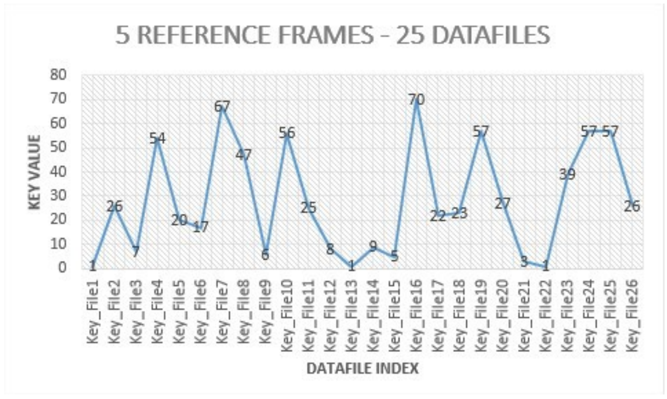


Figure 118 Key value obtained with 5 Reference Frames for 25 DataFiles for Vertical movement of the Device

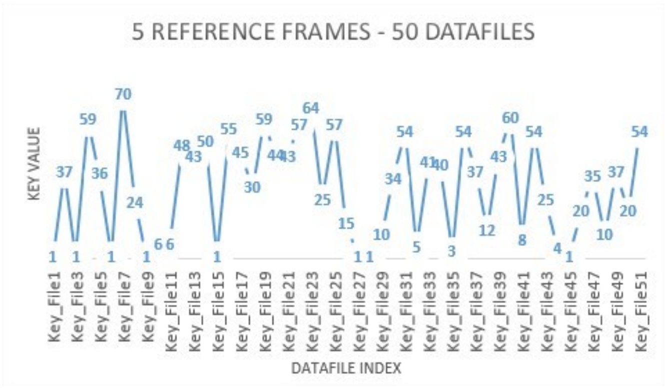


Figure 119 Key value obtained with 5 Reference Frames for 50 DataFiles for Vertical movement of the Device

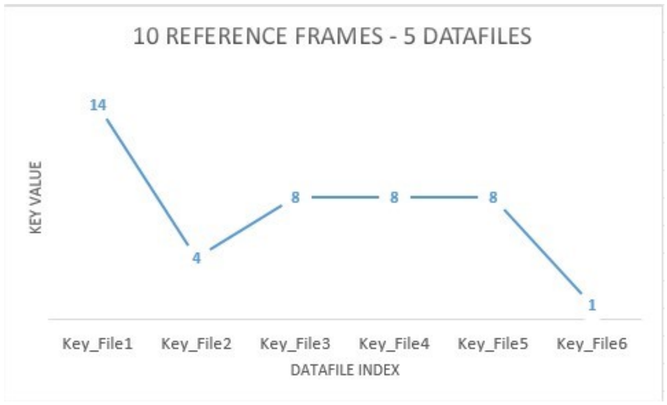


Figure 120 Key value obtained with 10 Reference Frames for 5 DataFiles for Vertical movement of the Device

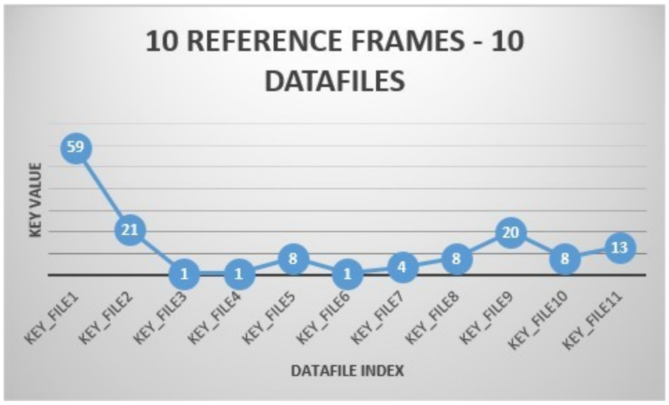


Figure 121 Key value obtained with 10 Reference Frames for 10 DataFiles for Vertical movement of the Device

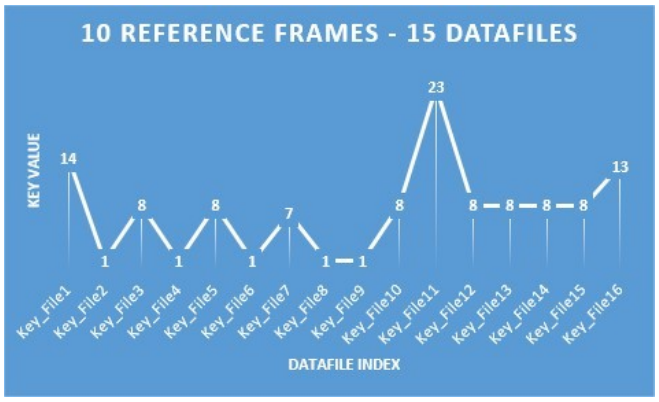


Figure 122 Key value obtained with 10 Reference Frames for 15 DataFiles for Vertical movement of the Device

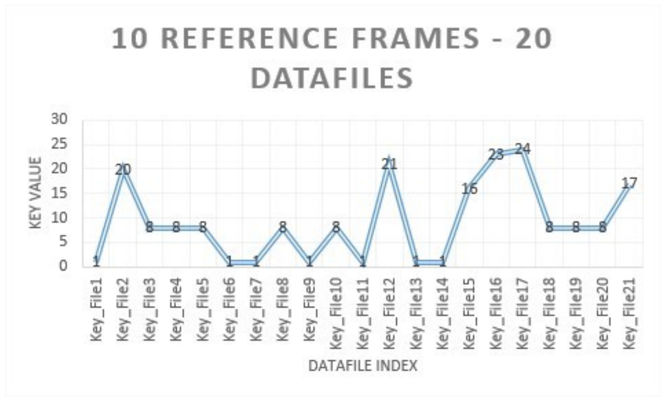


Figure 123 Key value obtained with 10 Reference Frames for 20 DataFiles for Vertical movement of the Device

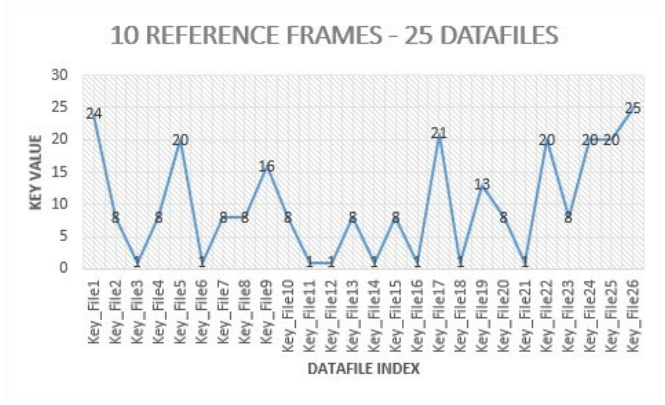


Figure 124 Key value obtained with 10 Reference Frames for 25 DataFiles for Vertical movement of the Device

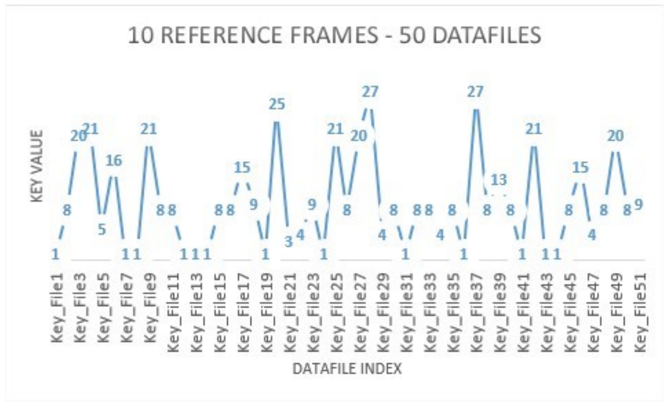


Figure 125 Key value obtained with 10 Reference Frames for 50 DataFiles for Vertical movement of the Device

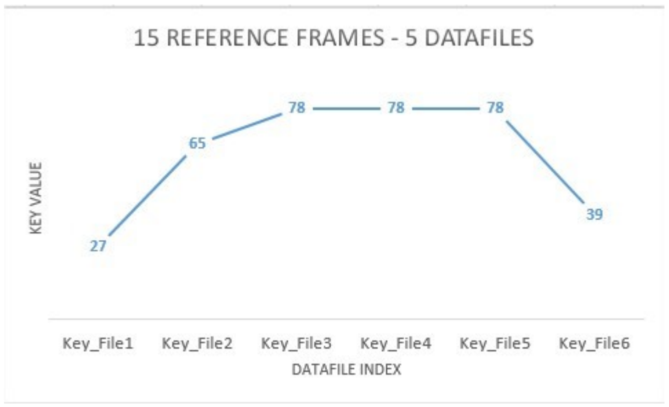


Figure 126 Key value obtained with 15 Reference Frames for 5 DataFiles for Vertical movement of the Device

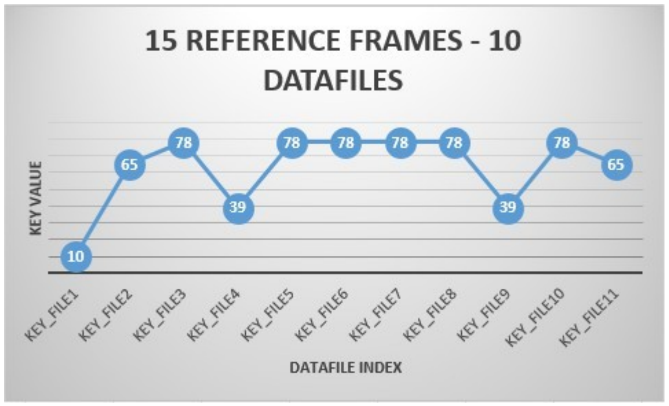


Figure 127 Key value obtained with 15 Reference Frames for 10 DataFiles for Vertical movement of the Device

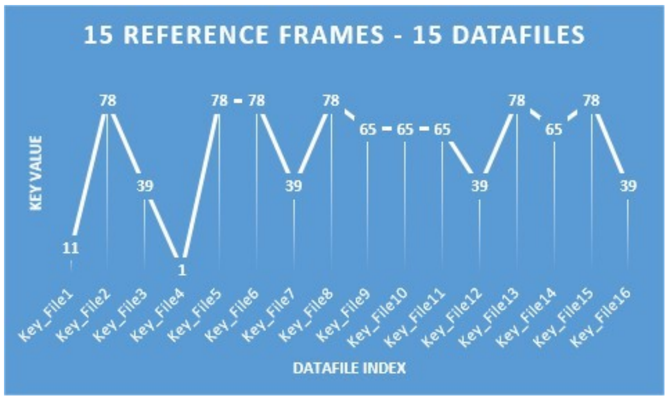


Figure 128 Key value obtained with 15 Reference Frames for 15 DataFiles for Vertical movement of the Device

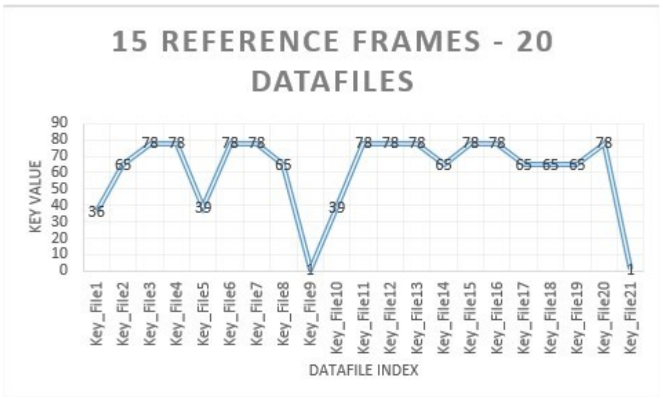


Figure 129 Key value obtained with 15 Reference Frames for 20 DataFiles for Vertical movement of the Device

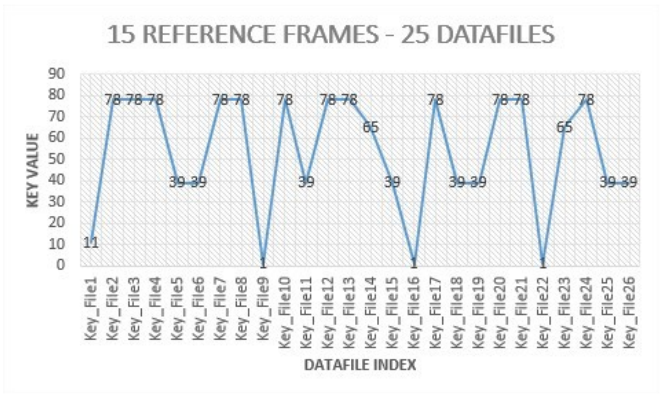


Figure 130 Key value obtained with 15 Reference Frames for 25 DataFiles for Vertical movement of the Device

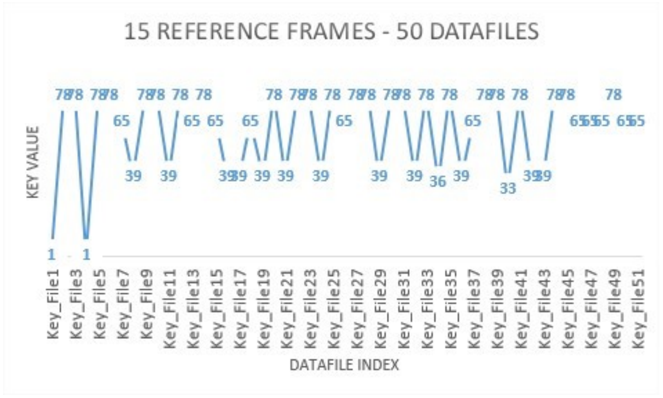


Figure 131 Key value obtained with 15 Reference Frames for 50 DataFiles for Vertical movement of the Device

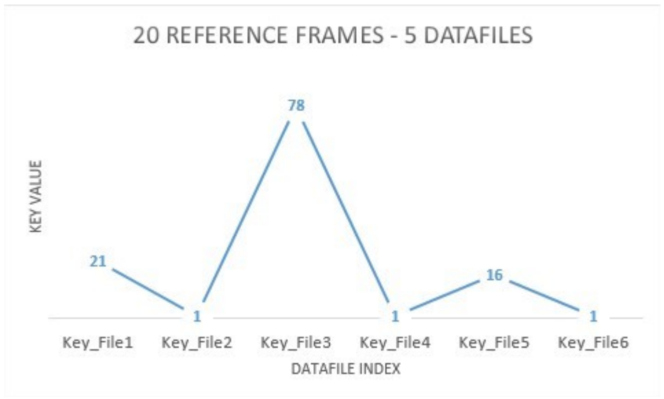


Figure 132 Key value obtained with 20 Reference Frames for 5 DataFiles for Vertical movement of the Device

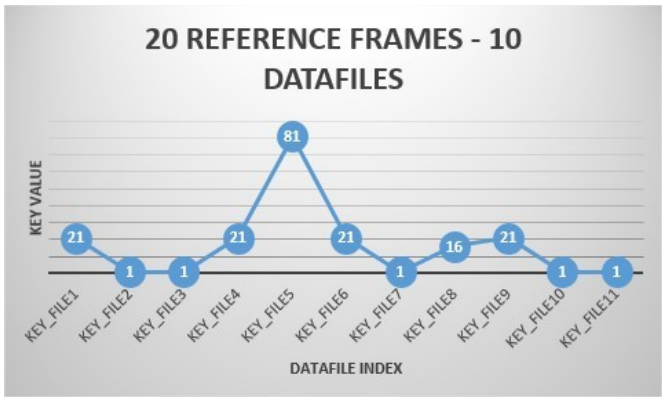


Figure 133 Key value obtained with 20 Reference Frames for 10 DataFiles for Vertical movement of the Device

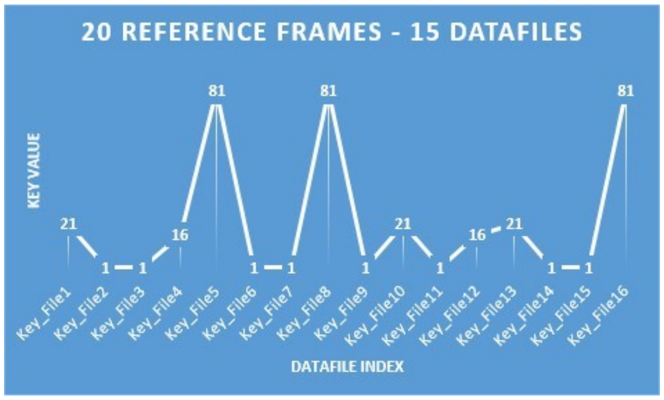


Figure 134 Key value obtained with 20 Reference Frames for 15 DataFiles for Vertical movement of the Device

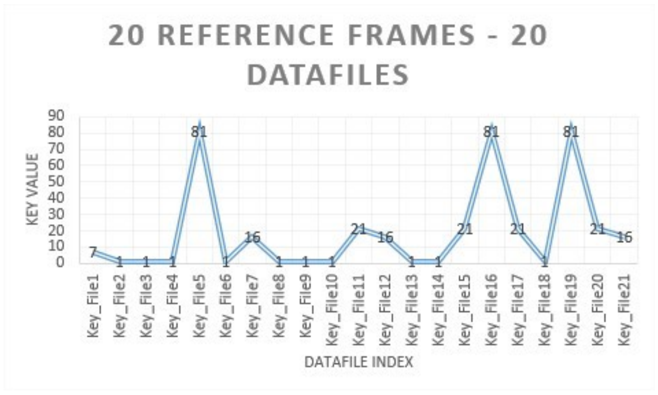


Figure 135 Key value obtained with 20 Reference Frames for 20 DataFiles for Vertical movement of the Device

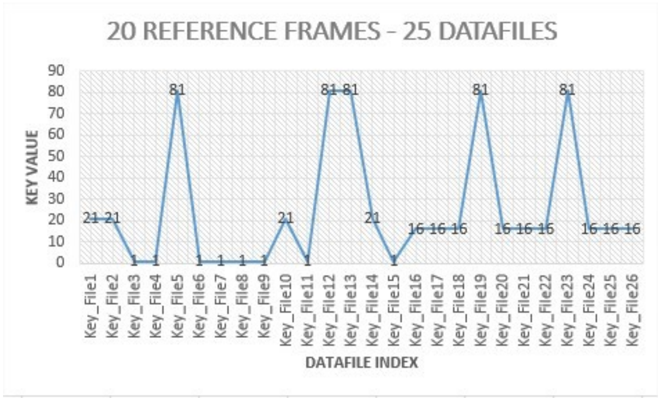


Figure 136 Key value obtained with 20 Reference Frames for 25 DataFiles for Vertical movement of the Device

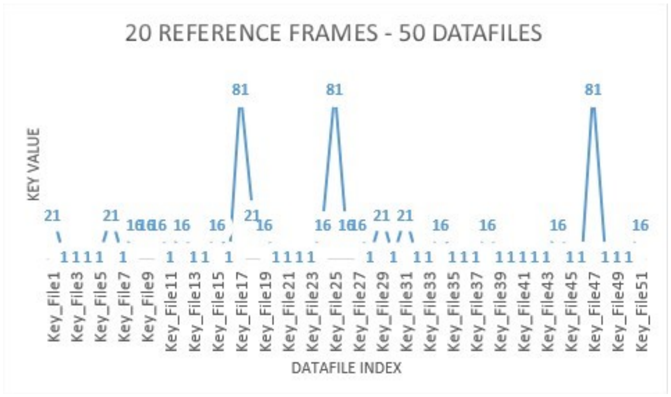


Figure 137 Key value obtained with 20 Reference Frames for 50 DataFiles for Vertical movement of the Device

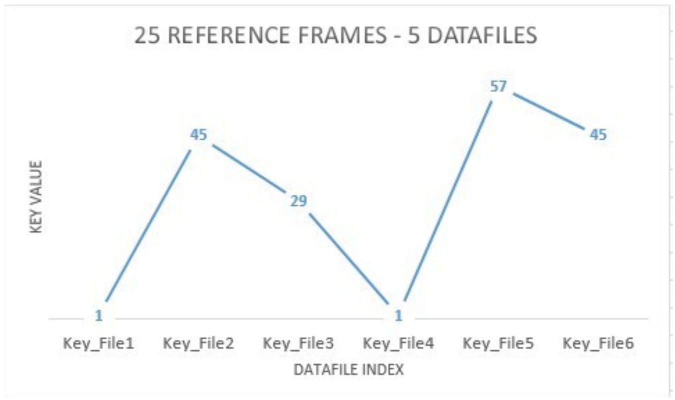


Figure 138 Key value obtained with 25 Reference Frames for 5 DataFiles for Vertical movement of the Device

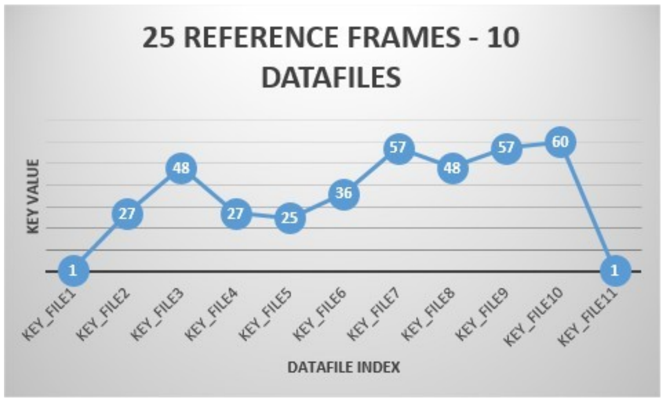


Figure 139 Key value obtained with 25 Reference Frames for 10 DataFiles for Vertical movement of the Device

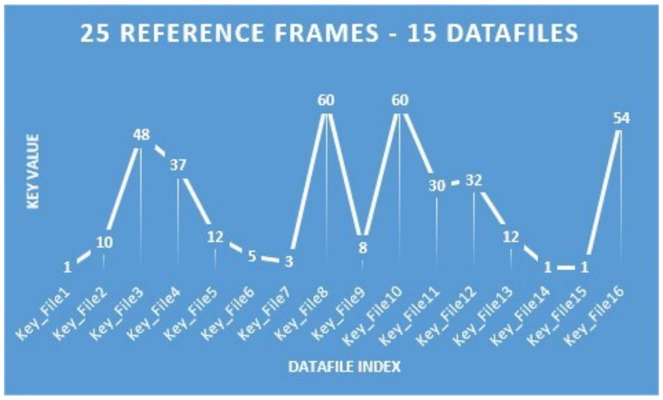


Figure 140 Key value obtained with 25 Reference Frames for 15 DataFiles for Vertical movement of the Device

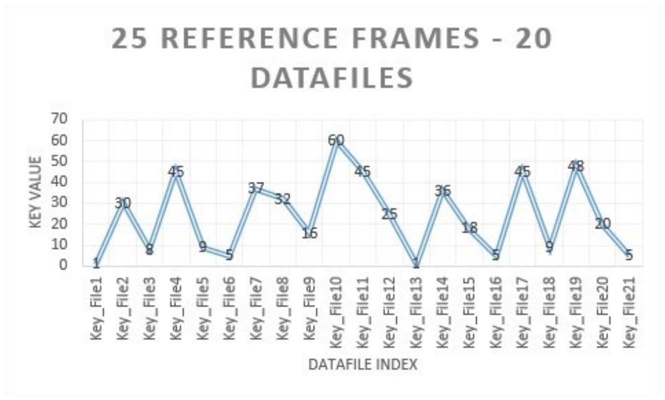


Figure 141 Key value obtained with 25 Reference Frames for 20 DataFiles for Vertical movement of the Device

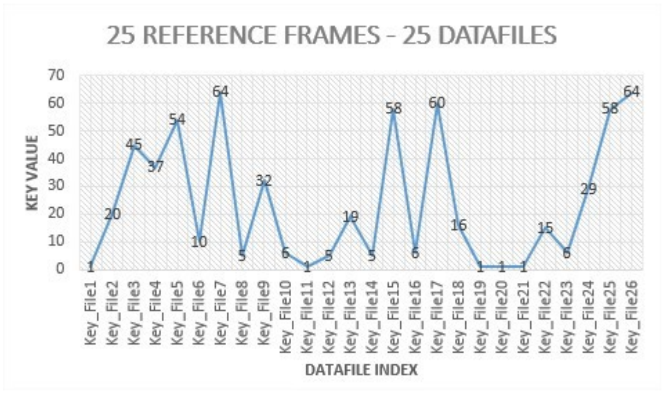


Figure 142 Key value obtained with 25 Reference Frames for 25 DataFiles for Vertical movement of the Device

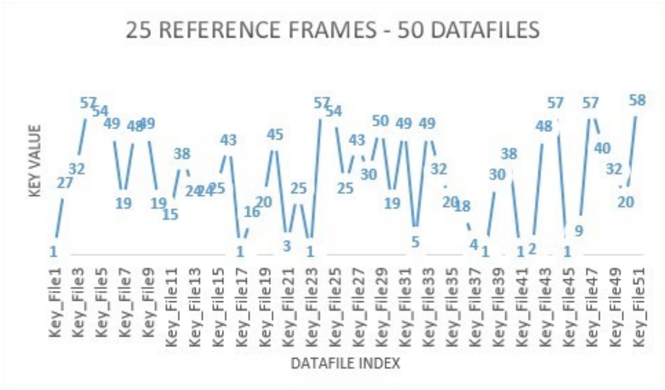


Figure 143 Key value obtained with 25 Reference Frames for 50 DataFiles for Vertical movement of the Device

APPENDIX C

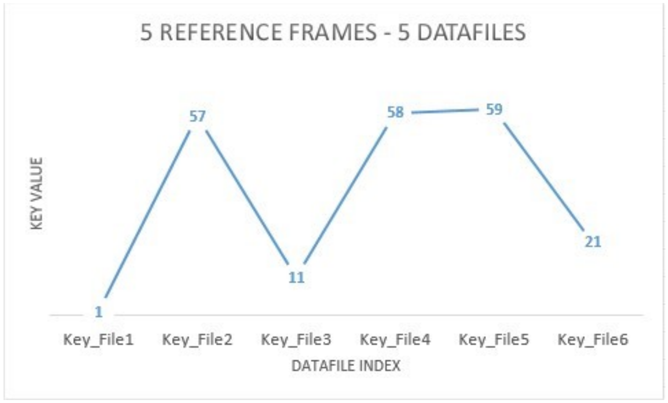


Figure 144 Key value obtained with 5 Reference Frames for 5 DataFiles when the Device is in Idle position

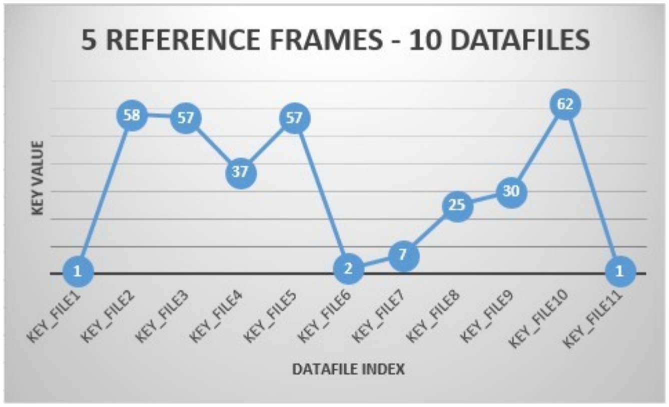


Figure 145 Key value obtained with 5 Reference Frames for 10 DataFiles when the Device is in Idle position

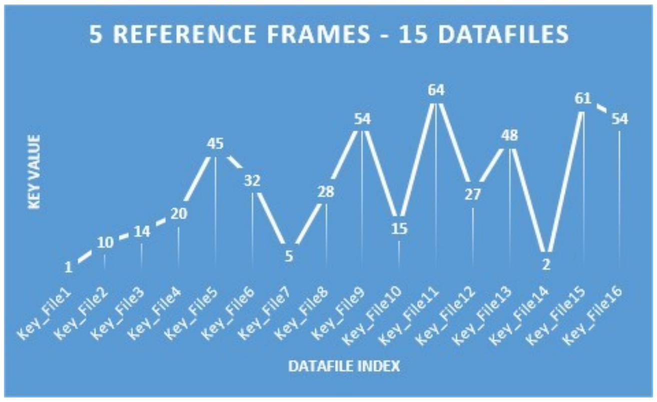


Figure 146 Key value obtained with 5 Reference Frames for 15 DataFiles when the Device is in Idle position

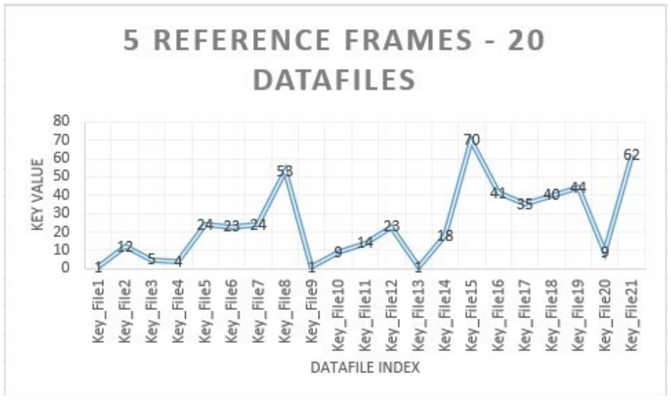


Figure 147 Key value obtained with 5 Reference Frames for 20 DataFiles when the Device is in Idle position

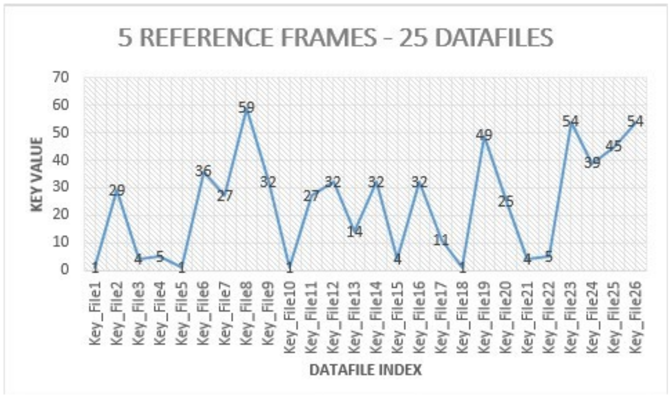


Figure 148 Key value obtained with 5 Reference Frames for 25 DataFiles when the Device is in Idle position

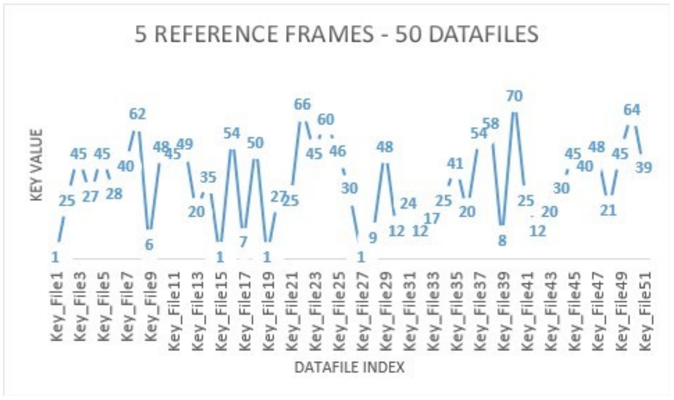


Figure 149 Key value obtained with 5 Reference Frames for 50 DataFiles when the Device is in Idle position

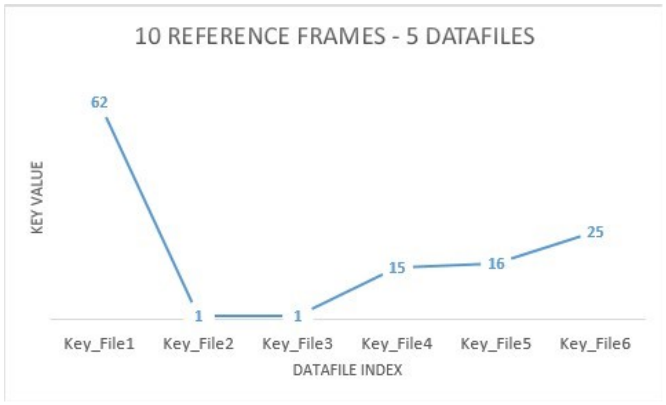


Figure 150 Key value obtained with 10 Reference Frames for 5 DataFiles when the Device is in Idle position

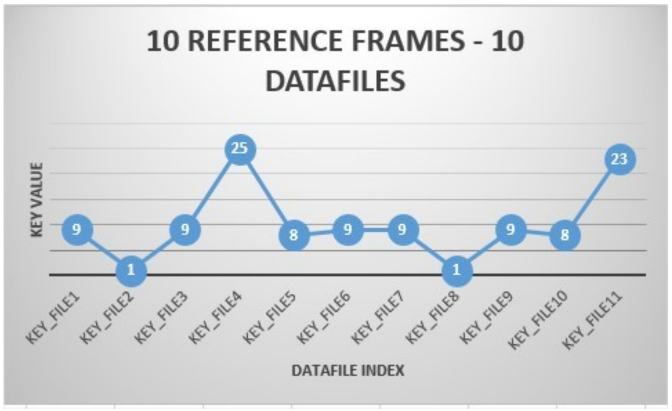


Figure 151 Key value obtained with 10 Reference Frames for 10 DataFiles when the Device is in Idle position

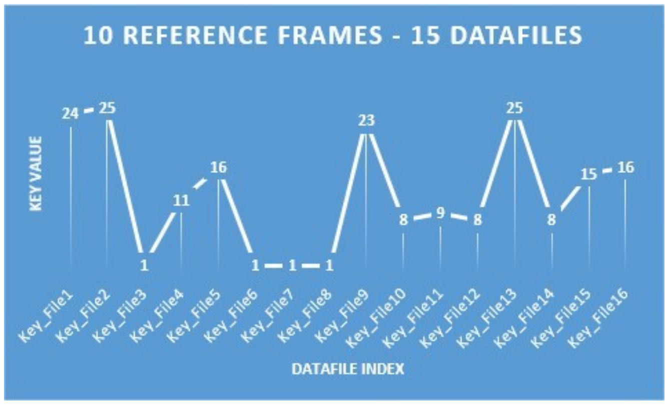


Figure 152 Key value obtained with 10 Reference Frames for 15 DataFiles when the Device is in Idle position

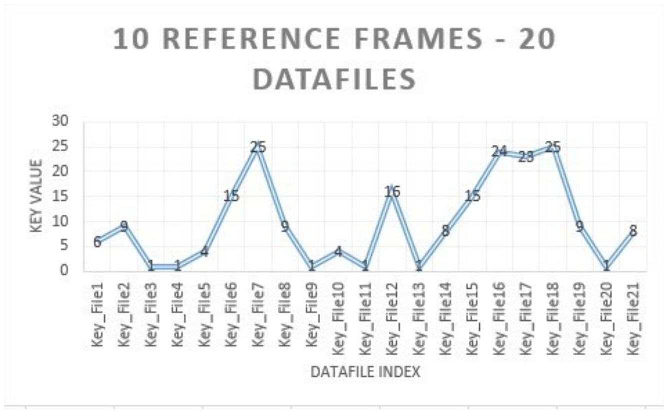


Figure 153 Key value obtained with 10 Reference Frames for 20 DataFiles when the Device is in Idle position

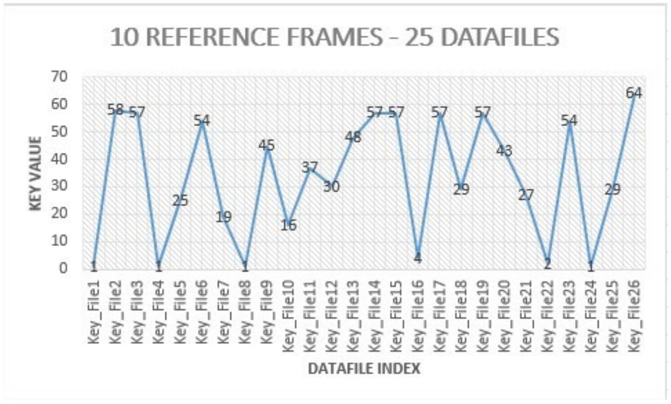


Figure 154 Key value obtained with 10 Reference Frames for 25 DataFiles when the Device is in Idle position

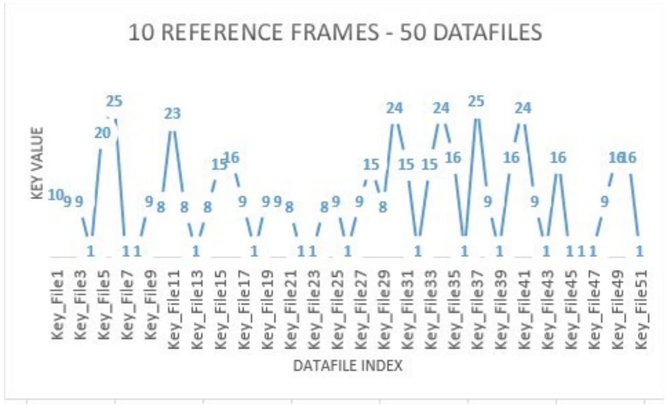


Figure 155 Key value obtained with 10 Reference Frames for 50 DataFiles when the Device is in Idle position

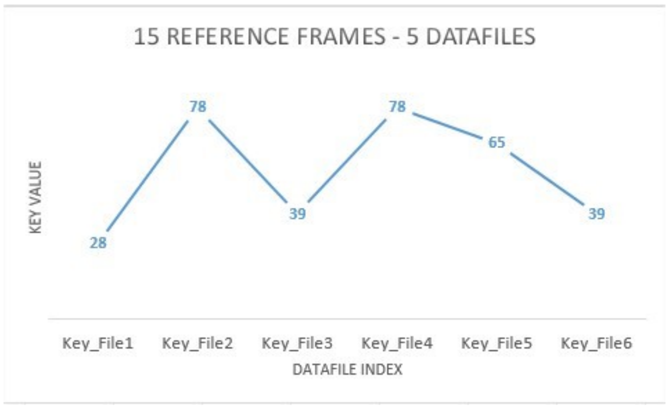


Figure 156 Key value obtained with 15 Reference Frames for 5 DataFiles when the Device is in Idle position

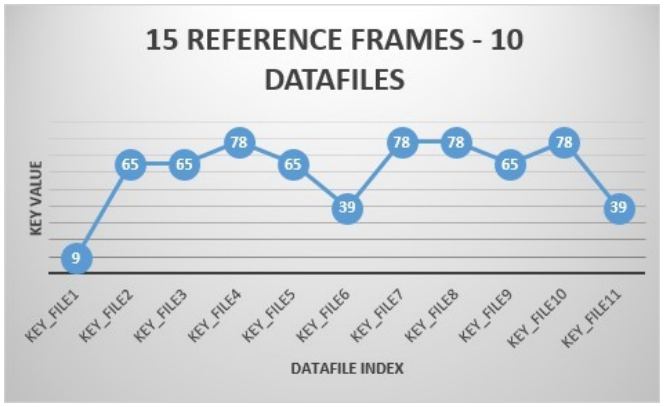


Figure 157 Key value obtained with 15 Reference Frames for 10 DataFiles when the Device is in Idle position

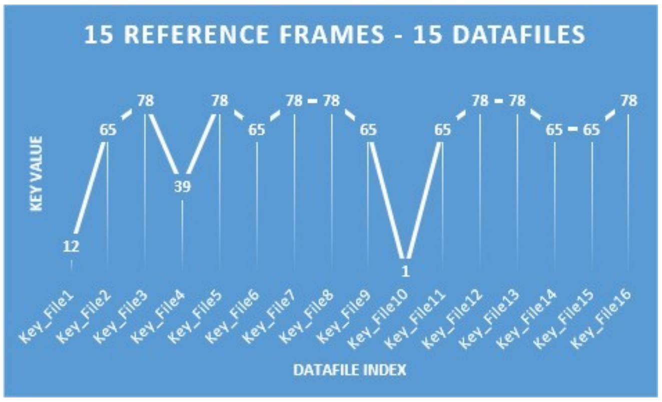


Figure 158 Key value obtained with 15 Reference Frames for 15 DataFiles when the Device is in Idle position

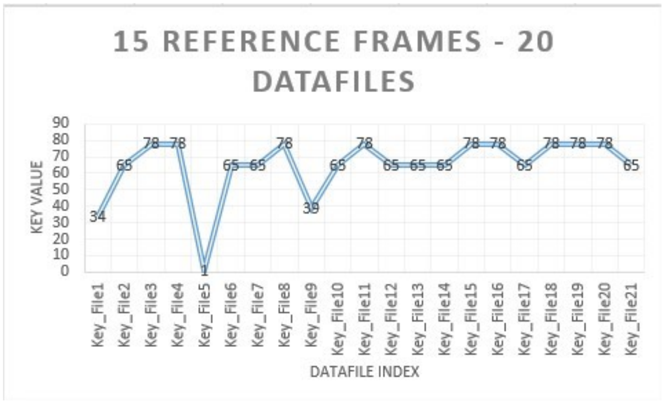


Figure 159 Key value obtained with 15 Reference Frames for 20 DataFiles when the Device is in Idle position

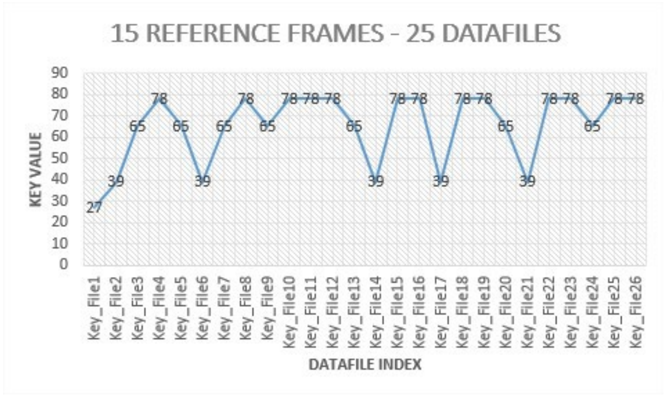


Figure 160 Key value obtained with 15 Reference Frames for 25 DataFiles when the Device is in Idle position

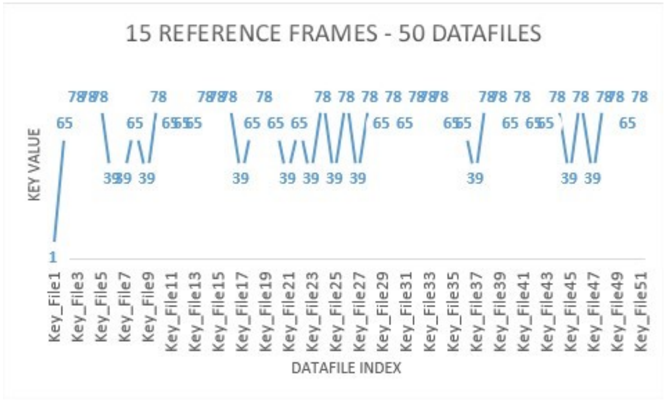


Figure 161 Key value obtained with 15 Reference Frames for 50 DataFiles when the Device is in Idle position

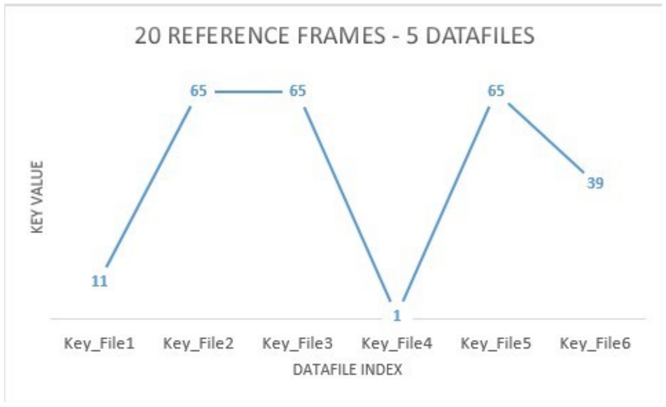


Figure 162 Key value obtained with 20 Reference Frames for 5 DataFiles when the Device is in Idle position

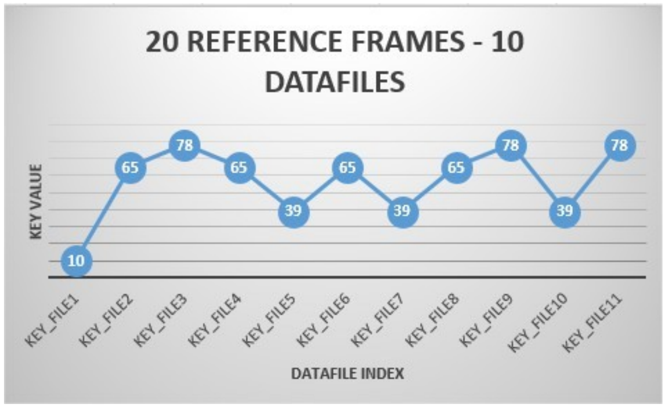


Figure 163 Key value obtained with 20 Reference Frames for 10 DataFiles when the Device is in Idle position

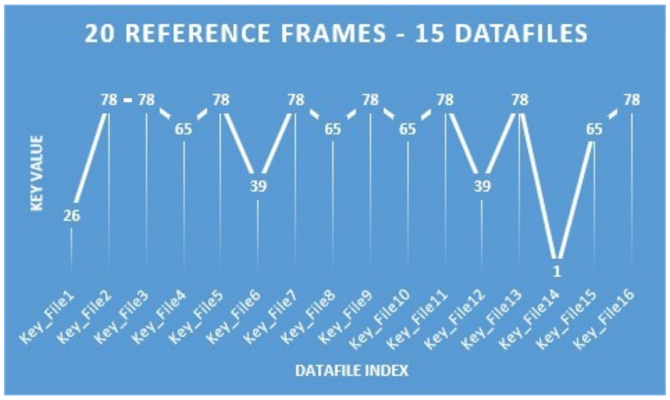


Figure 164 Key value obtained with 20 Reference Frames for 15 DataFiles when the Device is in Idle position

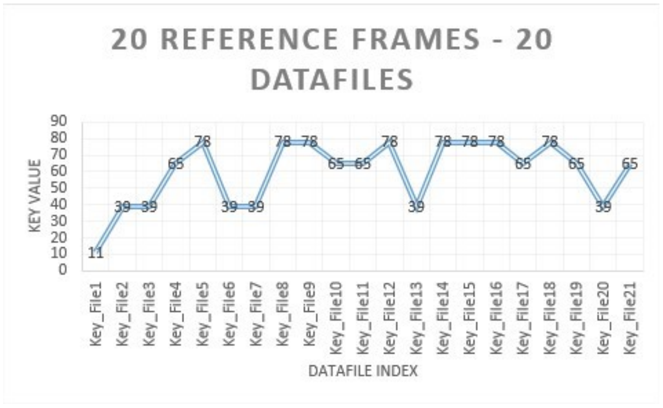


Figure 165 Key value obtained with 20 Reference Frames for 20 DataFiles when the Device is in Idle position

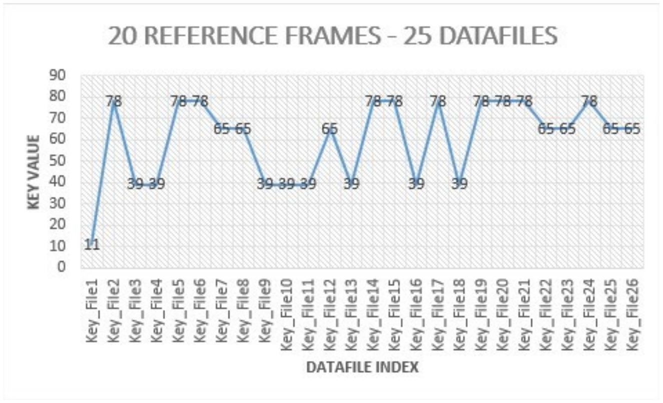


Figure 166 Key value obtained with 20 Reference Frames for 25 DataFiles when the Device is in Idle position

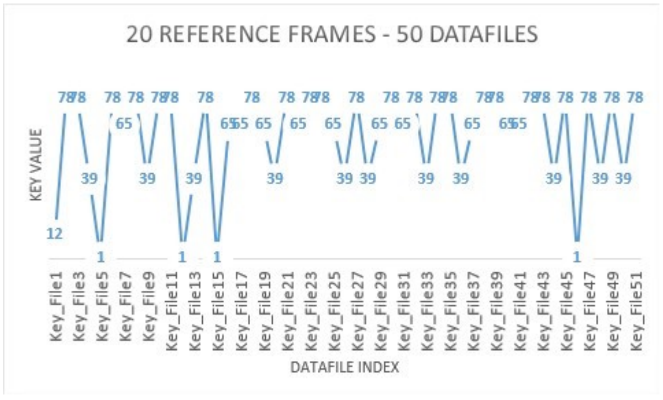


Figure 167 Key value obtained with 20 Reference Frames for 50 DataFiles when the Device is in Idle position

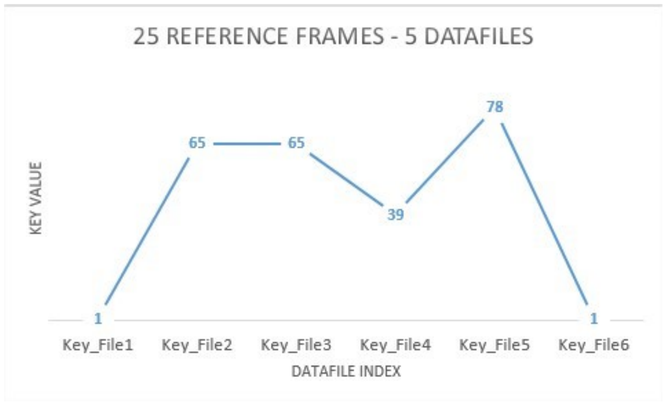


Figure 168 Key value obtained with 25 Reference Frames for 5 DataFiles when the Device is in Idle position

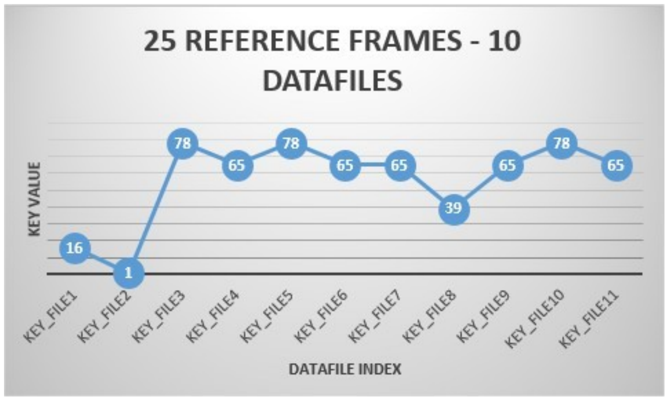


Figure 169 Key value obtained with 25 Reference Frames for 10 DataFiles when the Device is in Idle position

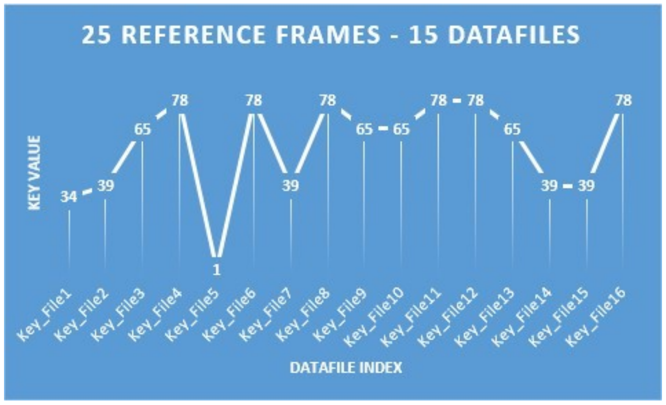


Figure 170 Key value obtained with 25 Reference Frames for 15 DataFiles when the Device is in Idle position

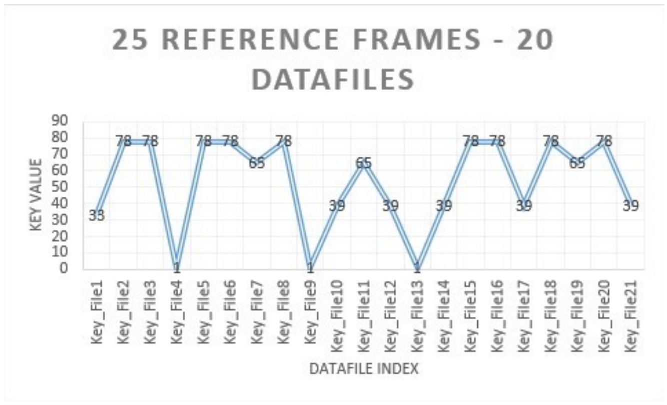


Figure 171 Key value obtained with 25 Reference Frames for 20 DataFiles when the Device is in Idle position

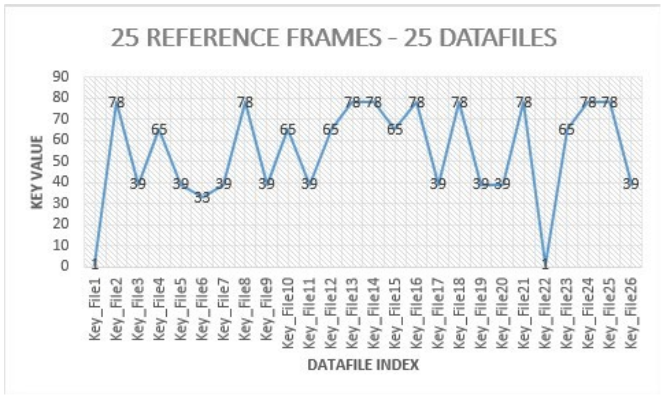


Figure 172 Key value obtained with 25 Reference Frames for 25 DataFiles when the Device is in Idle position

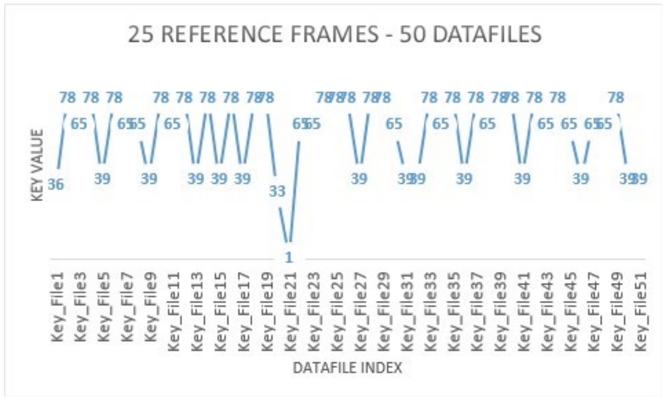


Figure 173 Key value obtained with 25 Reference Frames for 50 DataFiles when the Device is in Idle position

APPENDIX D

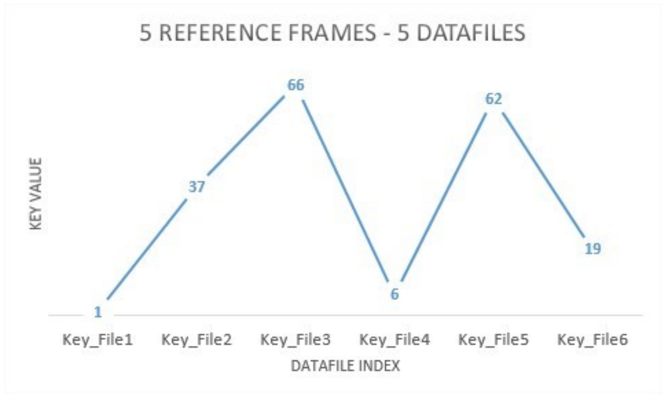


Figure 174 Key value obtained with 5 Reference Frames for 5 DataFiles when the Device is in trouser pocket while walking

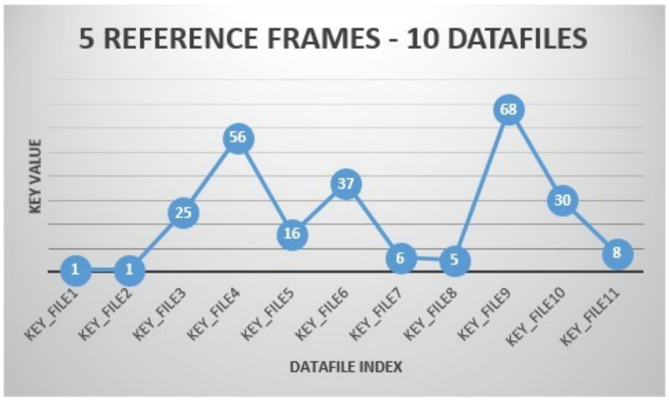


Figure 175 Key value obtained with 5 Reference Frames for 10 DataFiles when the Device is in trouser pocket while walking

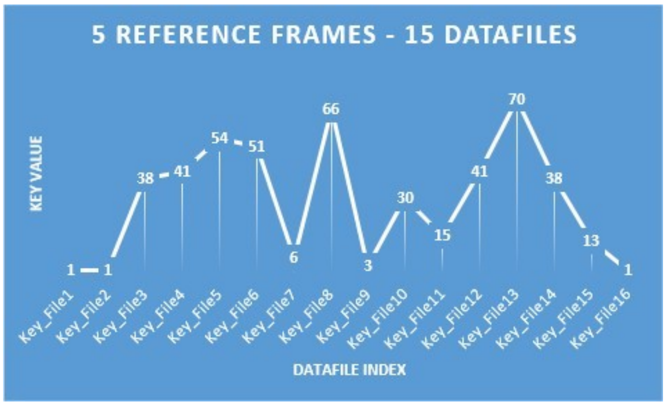


Figure 176 Key value obtained with 5 Reference Frames for 15 DataFiles when the Device is in trouser pocket while walking

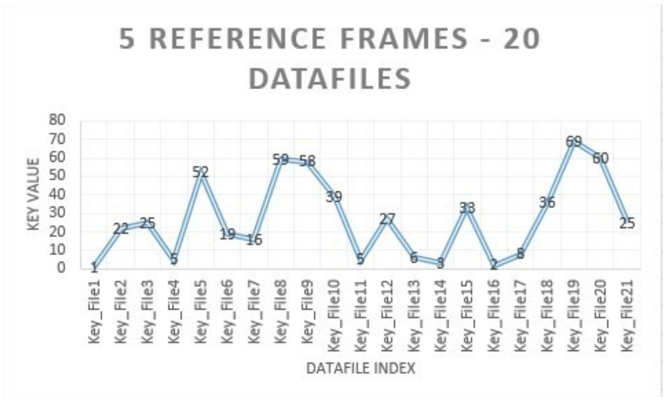


Figure 177 Key value obtained with 5 Reference Frames for 20 DataFiles when the Device is in trouser pocket while walking

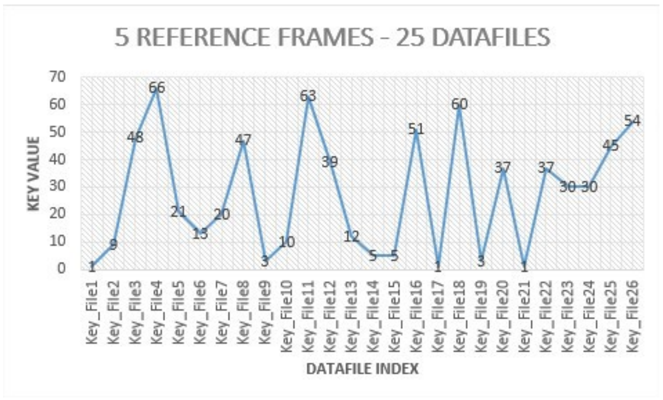


Figure 178 Key value obtained with 5 Reference Frames for 25 DataFiles when the Device is in trouser pocket while walking

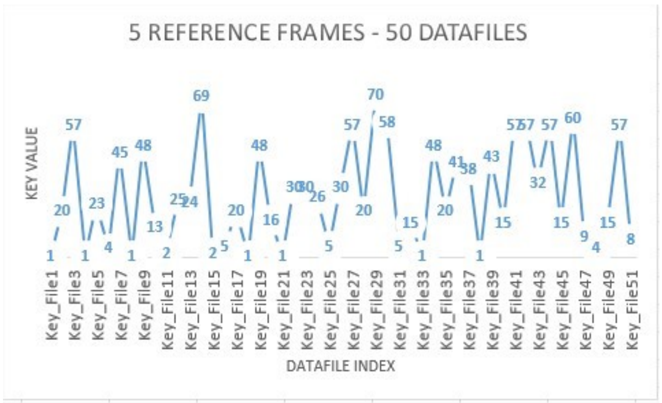


Figure 179 Key value obtained with 5 Reference Frames for 50 DataFiles when the Device is in trouser pocket while walking

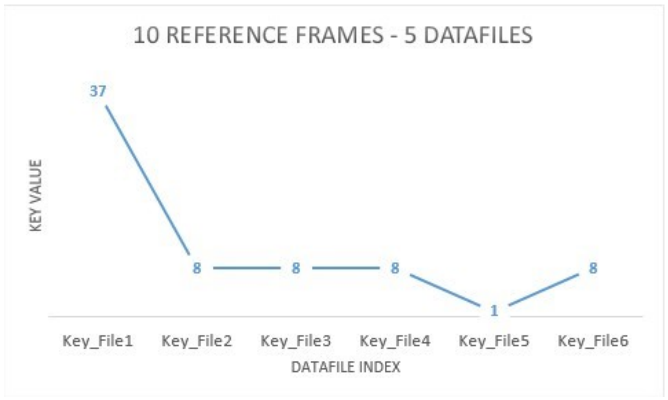


Figure 180 Key value obtained with 10 Reference Frames for 5 DataFiles when the Device is in trouser pocket while walking

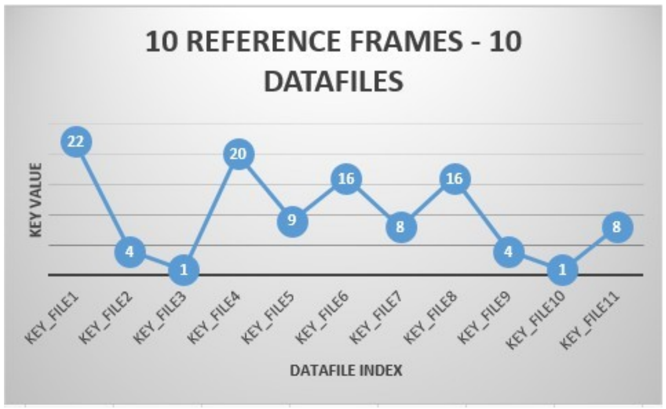


Figure 181 Key value obtained with 10 Reference Frames for 10 DataFiles when the Device is in trouser pocket while walking

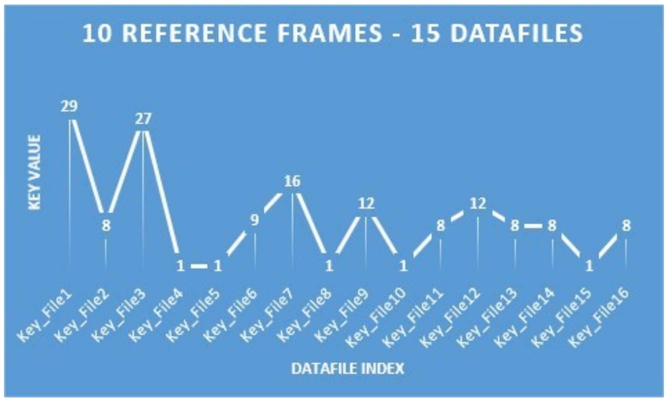


Figure 182 Key value obtained with 10 Reference Frames for 15 DataFiles when the Device is in trouser pocket while walking

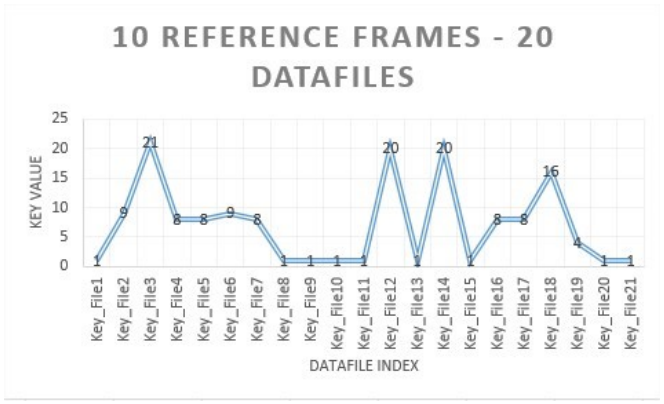


Figure 183 Key value obtained with 10 Reference Frames for 20 DataFiles when the Device is in trouser pocket while walking

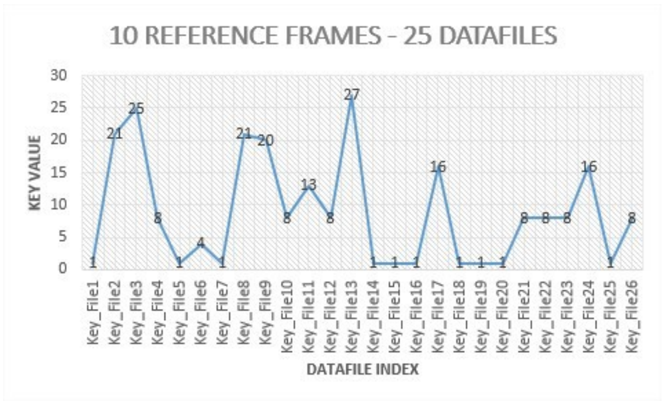


Figure 184 Key value obtained with 10 Reference Frames for 25 DataFiles when the Device is in trouser pocket while walking

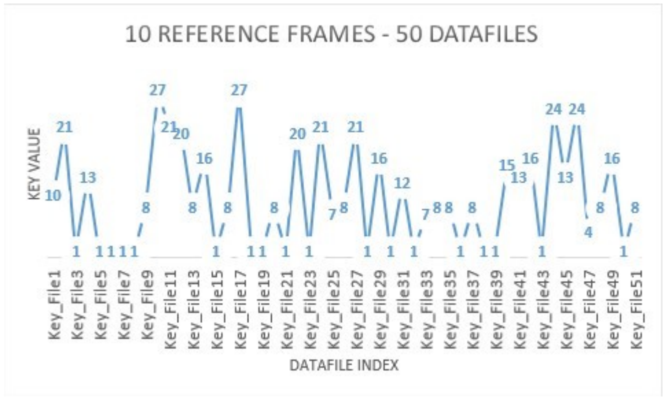


Figure 185 Key value obtained with 10 Reference Frames for 50 DataFiles when the Device is in trouser pocket while walking

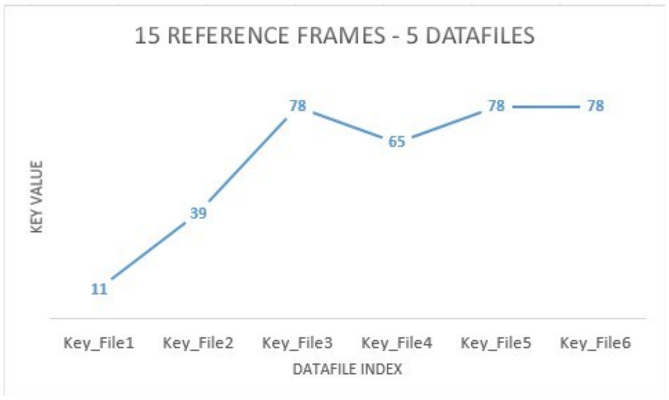


Figure 186 Key value obtained with 15 Reference Frames for 5 DataFiles when the Device is in trouser pocket while walking

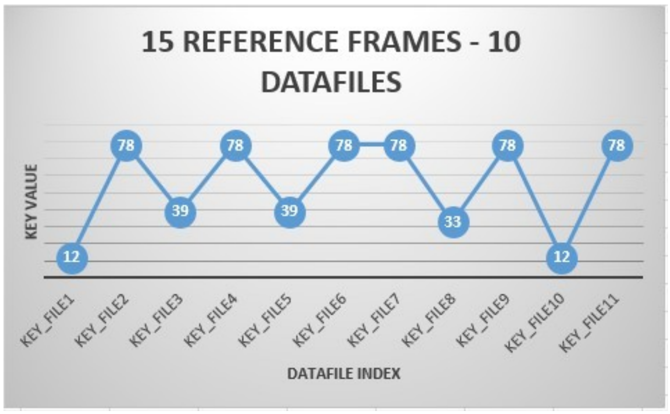


Figure 187 Key value obtained with 15 Reference Frames for 10 DataFiles when the Device is in trouser pocket while walking

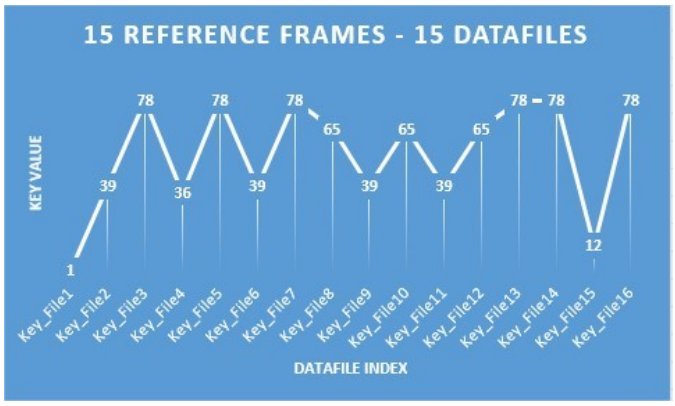


Figure 188 Key value obtained with 15 Reference Frames for 15 DataFiles when the Device is in trouser pocket while walking

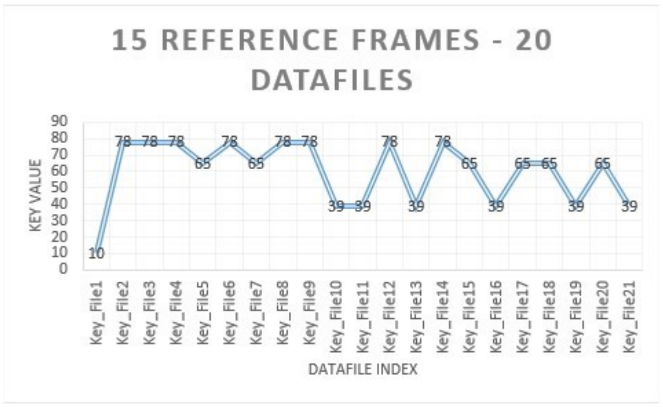


Figure 189 Key value obtained with 15 Reference Frames for 20 DataFiles when the Device is in trouser pocket while walking

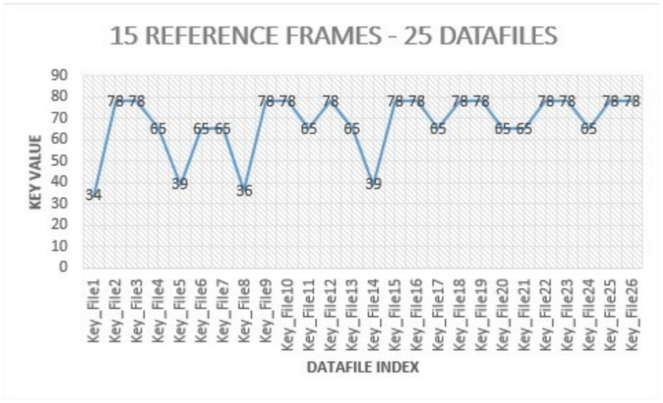


Figure 190 Key value obtained with 15 Reference Frames for 25 DataFiles when the Device is in trouser pocket while walking

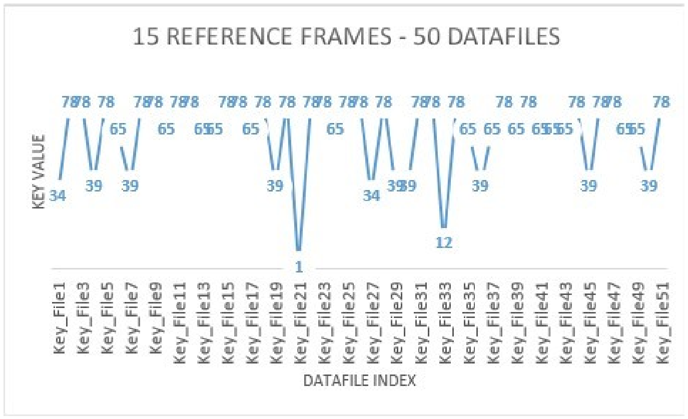


Figure 191 Key value obtained with 15 Reference Frames for 50 DataFiles when the Device is in trouser pocket while walking

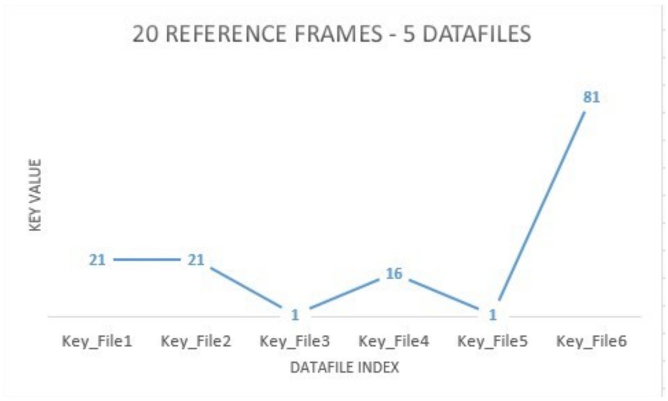


Figure 192 Key value obtained with 20 Reference Frames for 5 DataFiles when the Device is in trouser pocket while walking

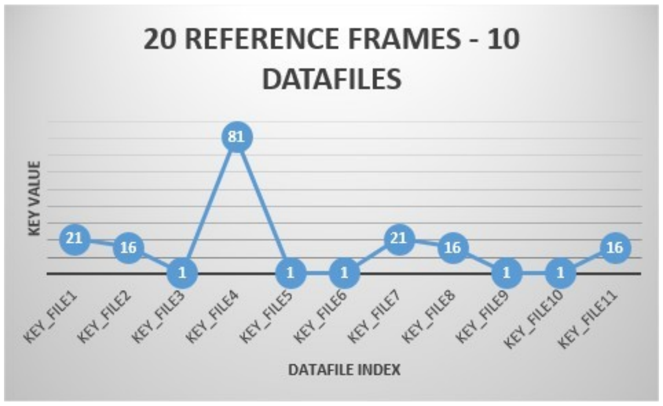


Figure 193 Key value obtained with 20 Reference Frames for 10 DataFiles when the Device is in trouser pocket while walking

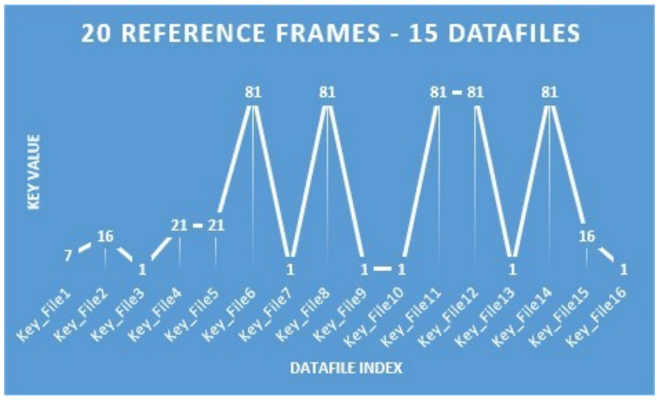


Figure 194 Key value obtained with 20 Reference Frames for 15 DataFiles when the Device is in trouser pocket while walking

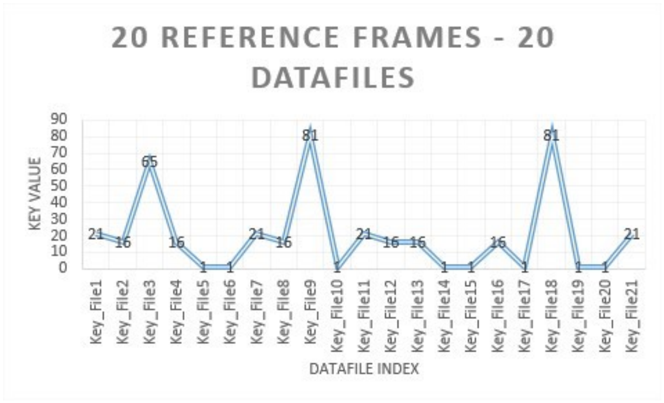


Figure 195 Key value obtained with 20 Reference Frames for 20 DataFiles when the Device is in trouser pocket while walking

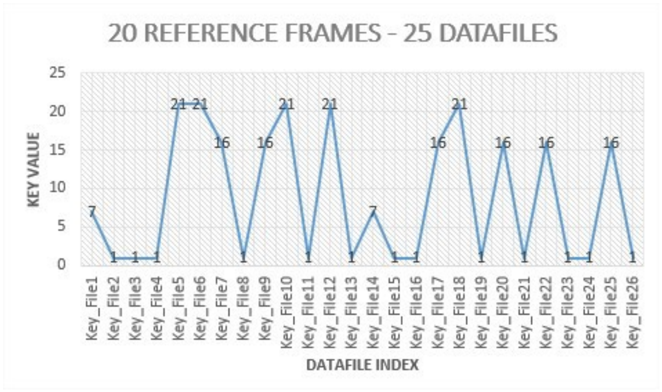


Figure 196 Key value obtained with 20 Reference Frames for 25 DataFiles when the Device is in trouser pocket while walking

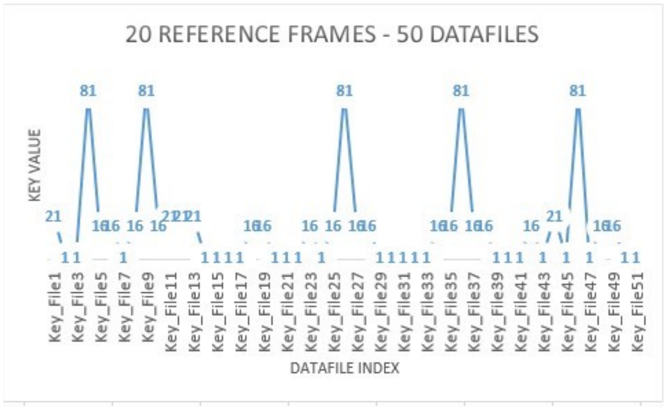


Figure 197 Key value obtained with 20 Reference Frames for 50 DataFiles when the Device is in trouser pocket while walking

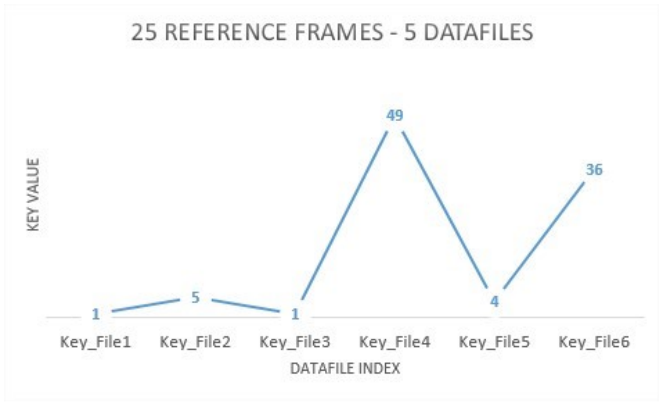


Figure 198 Key value obtained with 25 Reference Frames for 5 DataFiles when the Device is in trouser pocket while walking

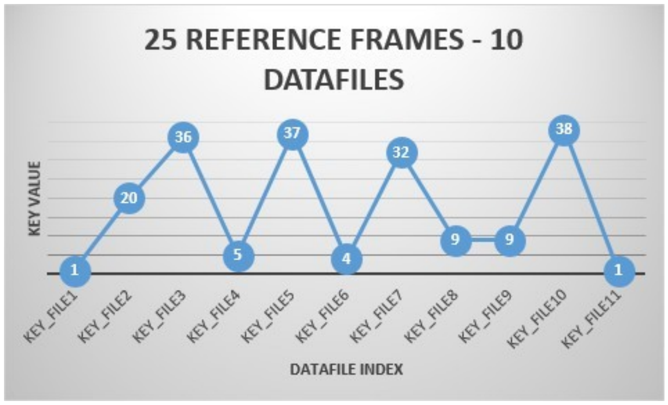


Figure 199 Key value obtained with 25 Reference Frames for 10 DataFiles when the Device is in trouser pocket while walking

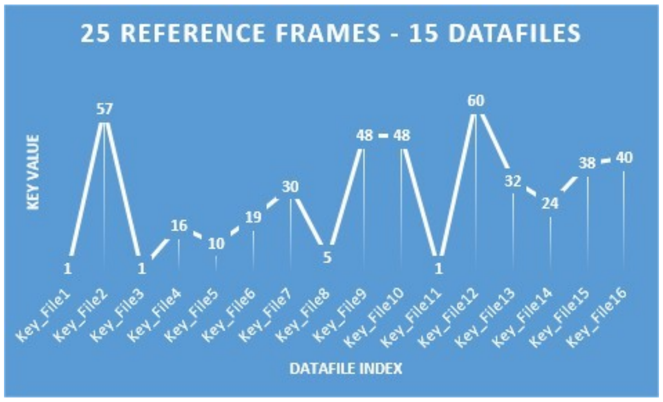


Figure 200 Key value obtained with 25 Reference Frames for 15 DataFiles when the Device is in trouser pocket while walking

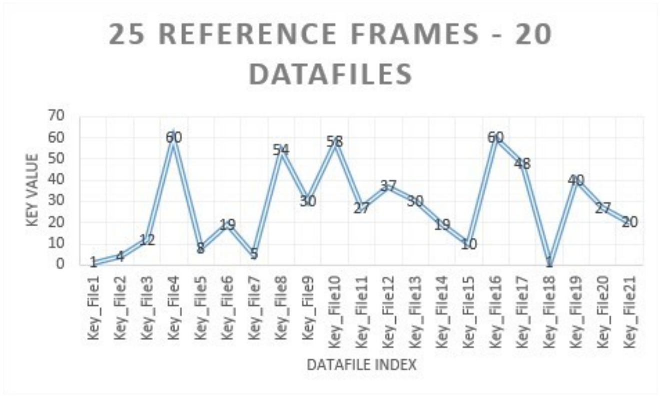


Figure 201 Key value obtained with 25 Reference Frames for 20 DataFiles when the Device is in trouser pocket while walking

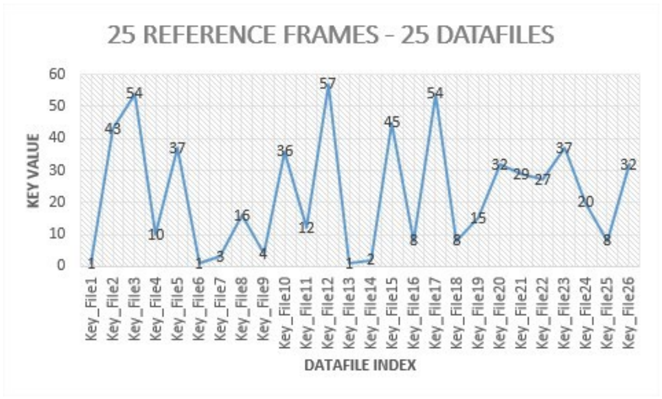


Figure 202 Key value obtained with 25 Reference Frames for 25 DataFiles when the Device is in trouser pocket while walking

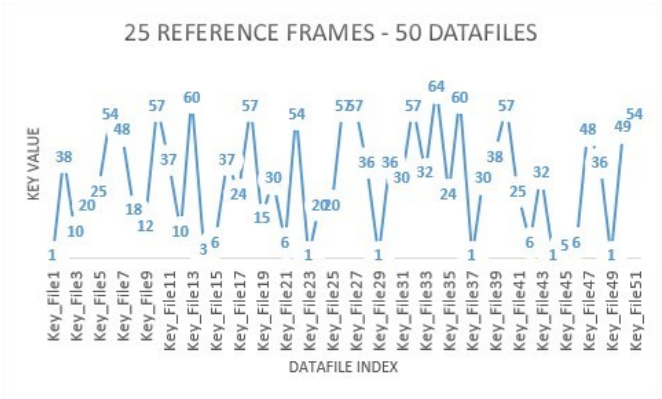


Figure 203 Key value obtained with 25 Reference Frames for 50 DataFiles when the Device is in trouser pocket while walking