SECURE, FAST, AND EFFICIENT HANDOFF

PROTOCOLS FOR WIRELESS MESH NETWORKS

by

NAIF ALAMRI

B.S., Taibah University, 2005

M.S., DePaul University, 2010

A dissertation submitted to the Graduate Faculty of the

University of Colorado at Colorado Springs

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2018
This dissertation for the Doctor of Philosophy degree by

NAIF ALAMRI

Has been approved for the

Department of Computer Science

by

C. Edward Chow, Chair

Jonathan Ventura

Sang-Yoon Chang

Yanyan Zhuang

Mark Wickert

Date ______________
Alamri, Naif (Ph.D. in Engineering, Focus in Security)

Secure, Fast, and Efficient Handoff Protocols for Wireless Mesh Networks

Thesis directed by Professor C. Edward Chow

ABSTRACT

In 2011, the IEEE 802.11s for wireless mesh networks (WMN) was introduced to provide the specifications and necessary changes to the MAC layer protocols which extend the network capabilities in establishing multi-hop and multi-path connections between wireless nodes within the same Mesh Basic Service Set (MBSS). However, this innovative and convenient approach to wireless communications is facing many challenges, most of which are influenced by the unique characteristics of WMN. These challenges include security, routing, and QoS.

In WMN, it is important to utilize fast handoff authentication protocols to guarantee a certain level of seamless roaming for the clients. However, the lack of a fast and reliable handoff protocol in the 802.11s standard is a major drawback especially in a technology designed to accommodate clients with high mobility. The use of authentication methods such as 802.1X/EAP for handoff authentication requires intensive computational power in devices with restricted resources and will cause a significant delay and QoS degradation. In this dissertation, we propose fast, and efficient handoff authentication protocols for WMN. First, a token-based authentication protocol is designed to provide localized handoff for clients with linear movement patterns. Second, a ticket-based authentication protocol is designed to provide global handoff within the boundaries of the mobility domain. We provide a performance assessment framework which is to compare our protocols to currently used protocols such as EAP-TLS and
EAP-PEAP. Performance analysis will prove that our protocols are 250 times faster than EAP-PEAP and 500 times faster than EAP-TLS. This improvement allows our protocols to provide seamless handoff and continuous operation for real-time applications with short-delay requirements such as VoIP, video conferences, and online games.
To my parents, my wife, and my family, the driving forces behind my success. Thank you for your dedication and your unconditional support throughout this tremendous journey.
ACKNOWLEDGMENT

To my passionate supporter, my lovely wife Mashael. No amount of thanks can ever express my gratitude. Your patience and devotion throughout the years have been the fuel of my motivation.

I would like to thank my wonderful parents for their unconditional love and encouragement. It's through the values you instilled in me that I was able to realize my potentials. This is yet another chapter of our success story.

I would like to express my sincere appreciation to my advisor and my mentor, Professor Edward Chow, whom I came to know and admire since I have met him during my first semester at UCCS. His guidance, thoughtfulness, and continuous support helped me achieve my research goals. It has been an honor to work with such an innovative mind for all these years.

I would like to thank the committee members, Professor Jonathan Ventura, Professor Sang-Yoon Chang, Professor Yanyan Zhuang, and Professor Mark Wickert. Their constructive feedback and valuable opinions helped shape the course of my research.

I would like to thank my faithful friends Amer Aljaedi and Abdelhamid Elgzil. Their kind support has been viable to my research effort especially during the preparation of my first published paper.

Finally, I would like to thank my employer, Taibah University, and the Saudi Arabian Cultural Mission (SACM) for the life-changing opportunity and the generous financial support.
# TABLE OF CONTENTS

CHAPTER

## I. INTRODUCTION

1.1 Overview of Wireless Mesh Networks ........................................... 1

1.2 Architectures of WMN .................................................................. 3

1.3 Components of WMNs ................................................................. 7

1.4 Routing Protocols and Metrics in WMNs ......................................... 8

1.5 Security in WMN .......................................................................... 22

1.6 Research Motivation ................................................................... 24

1.7 Dissertation Contributions ............................................................. 28

1.8 Dissertation Outline ..................................................................... 30

## II. RELATED WORK

2.1 Cryptographic Methods Used in Authentication ................................. 33

2.2 Current Wireless Security Schemes .................................................. 41

2.3 Related Wireless Mesh Networks Research ........................................ 62

## III. HALA: THE HANDOFF AUTHENTICATION LATENCY ASSESSMENT FRAMEWORK

3.1 Overview ..................................................................................... 85

3.2 Android Security System ............................................................... 91

3.3 iOS Security System .................................................................... 94

3.4 Experimental Setup of The HALA Framework on Android and iOS Devices. 98

3.5 Results and Analysis .................................................................... 101
IV. UFAP: ULTRA-FAST HANDBOFF AUTHENTICATION PROTOCOL........ 111

4.1 Introduction ........................................................................................................... 111
4.2 Design Principles for Handoff Protocols in WMN ........................................ 112
4.3 Handoff Protocol Design ...................................................................................... 113
4.4 Implementation and Experimental Setup .......................................................... 118
4.5 Performance Analysis .......................................................................................... 121
4.6 Security Analysis .................................................................................................. 124

V. GAP: THE GLOBAL HANDBOFF AUTHENTICATION PROTOCOL .......... 127

5.1 GAP Protocol Design ......................................................................................... 127
5.2 Handoff Process ................................................................................................... 129
5.3 Performance Analysis ......................................................................................... 132

VI. CONCLUSION .................................................................................................... 136

6.1 Conclusion ............................................................................................................ 136
6.2 Future Work .......................................................................................................... 138

REFERENCES .......................................................................................................... 139

APPENDIX A ............................................................................................................. 146

APPENDIX B ............................................................................................................. 164
LIST OF TABLES

TABLE

2.1 A comparison between DES, 3DES, AES, and Blowfish................................. 39

2.2 Equivalent security comparison by key size.................................................. 40

3.1 List of cryptographic methods and parameters can be utilized in the HALA
    framework........................................................................................................ 89

3.2 List of tested mobile devices............................................................................ 100

4.1 A detailed list of the devices used in this experiment...................................... 120

A.1 Execution time of 8-bytes secure random number generation on Android and iOS
    devices in nanoseconds .................................................................................. 146

A.2 Execution time of 16-bytes secure random number generation on Android and iOS
    devices in nanoseconds .................................................................................. 147

A.3 Execution time of 32-bytes secure random number generation on Android and iOS
    devices in nanoseconds .................................................................................. 148

A.4 Execution time of 64-bytes secure random number generation on Android and iOS
    devices in nanoseconds .................................................................................. 149

A.5 Execution time of SHA-2 hash code generation on Android and iOS devices in
    nanoseconds ....................................................................................................... 150

A.6 Execution time of SHA-2 hash code verification on Android and iOS devices in
    nanoseconds ....................................................................................................... 152

A.7 Execution time of AES-256 encryption on Android and iOS devices in nanoseconds
    ........................................................................................................................... 154
A.8 Execution time of AES-256 decryption on Android and iOS devices in nanoseconds ................................................................. 156

A.9 Execution time of RSA-2048 encryption on Android and iOS devices in nanoseconds ................................................................. 158

A.10 Execution time of RSA-2048 decryption on Android and iOS devices in nanoseconds ................................................................. 159

A.11 Execution time of digital signature generation on Android and iOS devices in nanoseconds ................................................................. 160

A.12 Execution time of digital signature verification on Android and iOS devices in nanoseconds ................................................................. 162
LIST OF FIGURES

FIGURE

1.1 Using an easily deployed Wireless Mesh Network (WMN) to increase the range of other networks such as WiMAX without the need for new constructions ............ 2
1.2 Infrastructure WMN architecture ........................................................................ 3
1.3 Client WMN architecture ..................................................................................... 5
1.4 Hybrid WMN architecture ................................................................................... 6
1.5 The header format of the RREQ packet in DSR ................................................. 15
1.6 The header format of the RREQ packet in AODV ........................................... 18
1.7 MPR nodes in OLSR ............................................................................................. 20
1.8 OLSR packet format ............................................................................................. 20
1.9 Authentication and key generation in 802.11s ................................................... 23
2.1 Symmetric key encryption and decryption ......................................................... 38
2.2 WEP packet format .............................................................................................. 41
2.3 TKIP packet format .............................................................................................. 44
2.4 802.1X authentication process .......................................................................... 48
2.5 802.11r key hierarchy ......................................................................................... 56
2.6 Over-the-Air Fast BSS Transition ...................................................................... 58
2.7 Over-the-DS Fast BSS Transition ....................................................................... 61
3.1 An overview of the HALA framework and its major components ................. 87
3.2 Keystore hierarchy in Android ........................................................................... 93
3.3 iOS key hierarchy ................................................................................................ 97
3.4 Execution time of sensitive data retrieval from the secure memory areas on Android and iOS devices in milliseconds ................................................................. 102
3.5 Execution time of 8-byte and 32-byte secure random number generation on Android and iOS devices in milliseconds ................................................................. 104
3.6 Execution time of SHA-2 hash generation and verification on Android and iOS devices in milliseconds .......................................................................................... 105
3.7 Execution time of AES-256 encryption and decryption on Android and iOS devices in milliseconds .......................................................................................... 107
3.8 Execution time of RSA-2048 encryption and decryption on Android and iOS devices in milliseconds .......................................................................................... 109
3.9 Execution time of digital signature generation and verification on Android and iOS devices in millisecond .................................................................................. 110
4.1 Handoff authentication in WMN using UFAP and EAP-TLS................................. 114
4.2 Token generation and handoff authentication using UFAP............................... 117
4.3 Experimental setup to test the performance of UFAP, EAP-TLS, and EAP-PEAP in a multi-hop WMN............................................................................................... 119
4.4 Number of authentication messages in UFAP, EAP-TLS, and EAP-PEAP......... 121
4.5 Runtime comparison between UFAP, EAP-TLS, and EAP-PEAP on different mobile devices............................................................................................................ 123
5.1 GAP Handoff ticket format.................................................................................. 128
5.2 Ticket generation and the subsequent handoff process using GAP.................. 131
5.3 Runtime comparison between UFAP and GAP................................................... 135
5.4 Runtime comparison between UFAP, GAP, EAP-TLS, and EAP-PEAP on different mobile devices ................................................................. 135

B.1 The first screen in the application provides the ability to run all the cryptographic methods at once. These methods are run independently and consecutively and the result from the run of each method is not affected by the others. ......................... 165

B.2 In this screen, the user can generate secure random data by choosing needed data size from the list (8, 16, 32, 64, 128, and 256 bytes). The execution-times from 20 consecutive runs are displayed in nanoseconds. The average time is displayed in the upper box. ................................................................. 166

B.3 In this screen, the user can generate and verify hash codes using the SHA-2 hash function. Different data sizes can be added easily to the code. In this implementation, we used a 64-byte test file. The user can choose whether to generate a new hash code from a file or to verify a hash code received with another file. The execution-times from 20 consecutive runs are displayed in nanoseconds. The average time is displayed in the lower box. ................................................................. 167

B.4 In this screen, the user can encrypt or decrypt existing files using AES-256. Different data sizes can be added easily to the code. In this implementation, we used a 64-byte test file. The user can choose to run encryption or decryption. The execution-times from 20 consecutive runs are displayed in nanoseconds. The average time is displayed in the lower box. ................................................................. 168

B.5 In this screen, the user can encrypt or decrypt existing files using RSA-2048. Different data sizes can be added easily to the code. In this implementation, we used a 64-byte test file. The user can choose to run encryption or decryption. The
execution-times from 20 consecutive runs are displayed in nanoseconds. The average time is displayed in the lower box.

B.6 In this screen, the user can generate or verify digital signatures. The digital signature in this implementation is a signed hash code generated using SHA-2 and RSA-2048. Different data sizes can be added easily to the code. In this implementation, we used a 64-byte test file. The user can choose to run digital signature generation or verification. The execution-times from 20 consecutive runs are displayed in nanoseconds. The average time is displayed in the lower box.
CHAPTER I.
INTRODUCTION

1.1 Overview of Wireless Mesh Networks

Wireless Mesh Networking (WMN) has gained a lot of attention recently as it is designed to improve user experience and overall performance by introducing multi-hop and multi-path networking into the wireless ecosystem. The IEEE 802.11s standard [1], introduced in 2011, is an amendment to the IEEE 802.11 standard for wireless local area networks produced by Task Group (TGs). It contains the specifications and necessary changes to the MAC layer protocols that extend the network capabilities in establishing multi-hop and multi-path connections between wireless nodes within the same Mesh Basic Service Set (MBSS). The standard provides detailed specifications on how the MAC layer and the physical layer should operate to achieve a reliable and efficient mesh connectivity. In this case, the network consists of two or more mesh nodes connected using peer-to-peer wireless mesh links.

At one end of the spectrum, WMN is similar to wireless sensor networks and wireless ad-hoc networks since it is designed to be self-configured, self-organized, and self-healed. At the other end, the design of its elements promotes the increase of cost-effectiveness, deployment agility, scalability, and fault tolerance. It is evident that WMN is a more feasible alternative to other wireless networking technologies as it reduces the cost of deployment, infrastructural upgrades, operation costs, network management, and maintenance. Current applications of WMNs include last-mile broadband network access for homes, backbone of enterprises, and range extension for existing networks such as Wi-
Fi and WiMAX. In this case, WMN can be used to extend the range of other networks such as WiMAX to provide broadband wireless services to end users in remote areas outside the range of the WiMAX base stations [5]. The WiMAX base station will be connected with one of the mesh routers via a wired or wireless link. Figure 1 illustrates how WMNs can be used to extend the range of WiMAX to provide broadband access to end-users. An example of current WMN projects is Operation Virtual Shield (OVS), a city-wide surveillance system deployed in Chicago, which provides services such as facial recognition, and biological, chemical, and radiological substance detection [8]. Another example is PanOULU, a city-wide wireless mesh network built in the city of Oulu, Finland which provides easy and convenient access to the Internet throughout the city [67].

![Diagram of WiMAX and WMN integration](image)

Figure 1.1 Using an easily deployed Wireless Mesh Network (WMN) to increase the range of other networks such as WiMAX without the need for new constructions
1.2 Architectures of WMN

There are three types of WMNs architecture:

1) *Infrastructure WMN*. Also known as the Backbone WMN. In this approach, the network consists of dedicated mesh routers which allows the clients to communicate over wired and wireless links. The mesh routers are connected to each other using mesh links that are designed to be self-configured, self-organized while using the same routing protocol. The mesh routers can be connected to the Internet and other networks using mesh gateway routers. The clients can only access the network via legitimate mesh access points. WMNs are designed to be integrated with other wired and wireless networks using bridging or gateway capabilities. Figure 2 demonstrates how mesh routers form the wireless mesh backbone which is used here to connect different types of networks and clients to the Internet.

![Figure 1.2 Infrastructure WMN architecture](image)
2) **Client WMN.** Similar to conventional ad hoc networks, the network in this architecture consists of a number of clients with no dedicated mesh routers or authentication servers. In this case, the clients are connected using peer-to-peer wireless links and have equal rights and privileges. In client WMN, it is the responsibility of the clients to perform the routing, configuration, and maintenance of the mesh. The disadvantage of this type is the increase workload and energy consumption on the mobile devices with limited recourses. This is particularly a concern because most of the devices used in MANETs and WMNs depend on batteries [6]. For authentication purposes, the 802.11s standard suggests the utilization of the Simultaneous Authentication of Equals (SAE), a new shared-password authentication protocol designed for the client WMN architecture. Figure 1.3 demonstrates the multi-hop and multi-hop connections between the nodes in the client WMN architecture.
3) *Hybrid WMN.* This approach combines the characteristics of both the infrastructure and the client architectures. It takes advantage of properties of the client architecture to improve the connectivity and the throughput by supporting ad hoc networking which would increase the number of potential routes between the nodes. On the other hand, it benefits from the capability of the infrastructure architecture to connect with the Internet and with other radio technologies such as Wi-Fi, WiMAX, and cellular. Figure 4 depicts a hybrid WMN architecture where the mesh routers form the infrastructure WMN (backbone) while some mesh clients from the client WMN (adhoc).
The advantages of the Hybrid WMNs include:

- The capability of self-forming, self-configuring, self-organizing, and self-healing.
- Multi-hop and multi-path ad hoc network with an infrastructure provided by the mesh routers.
- The mesh routers, some of them are stationary while others have minimal mobility, perform the routing and some other network related functions which decreases the load on the mesh clients.
- It allows the integration and interoperability with other wireless networks.
- It takes into consideration the different energy consumption requirements of mesh routers and mesh clients.

Figure 1.4 Hybrid WMN architecture
1.3 Components of WMNs

There are three components of WMNs which are mesh routers, mesh clients, and mesh gateway routers.

1) **Mesh routers.** The mesh routers form the backbone of the network. These routers perform the routing between the clients as well as between the clients and the gateway routers. It is the mesh routers responsibility to maintain the self-forming, self-configuring, and self-healing features of WMN. In a normal setup, mesh routers would have limited mobility and steady power sources. Mesh routers can improve the bandwidth of the network by using multi-channel single-radio (MCSR) or multi-channel multi-radio (MCMR) technologies.

2) **Mesh clients.** Mesh clients are highly mobile devices with power constraints. This includes the end-user devices with wireless capabilities such as laptops, smartphones, tablets, and high-tech vehicles. Most of these devices are equipped with a single-radio. Moreover, mesh clients can be responsible for making the routing decisions as well as other networking functions in the WMN client architecture and the WMN hybrid architecture.

3) **Mesh gateway routers.** Mesh gateway routers are the backhaul of WMNs which provide access to the Internet and other networks. Mesh gateways allow the interoperability between WMNs and other wired and wireless networks such as Ethernet, Wi-Fi, WiMAX, cellular, and mobile ad hoc wireless networks (MANETS).
1.4 Routing Protocols and Metrics in WMNs

Routing is one of the most essential parts of networking. It is responsible for the efficient delivery of data units between end hosts which reside in different networks. The set of rules used in path setup, selection, maintenance, and delivery of data units is called a routing protocol.

In ad-hoc network, the path selection and routing processes are affected by the high mobility nature of the nodes. The life time of a route can be very short in comparison to wired networks. Nodes can lose connectivity due to signal interference or because a node along the path has moved from one location to another.

Routing protocols in ad-hoc networks can be classified into three categories:

1) Proactive routing protocols. Also known as table-driven protocols. In proactive routing, each node generates a routing table which contains fresh information regarding all routes from that node to all other nodes in the network. The routing table must be created once a new node joins the network. The new node must obtain and maintain fresh routes to all nodes in the network prior to sending data units to any other node. Each record in the routing table contains the next hop to the desired node and the cost of that route. Using proactive routing protocols, nodes can start transmitting data immediately because they already have all the routes information needed. However, proactive routing protocols perform poorly in large network because nodes will have to keep one or more records of each other node in their routing table which requires large memory capacity. This is not an applicable option for mobile devices with limited memory capabilities. Moreover, maintaining fresh routes in the routing
table requires the periodic exchange of updates between the nodes. This process creates routing overhead and reduces the network capacity by consuming more bandwidth. Some examples of proactive routing protocols are Destination Sequenced Distance Vector (DSDV) [55] and Optimized Link State Routing (OLSR) [58].

2) Reactive routing protocols [62]. Also known as On-demand protocols. Reactive routing protocols establish and maintain route information to a certain destination only when the source node attempts to communicate with that destination. The routing protocol will not have a global view of the network. Routes are established by flooding route request packets to all nodes in the network and waiting for the proper reply. The same process is triggered upon the revelation of a failed link along the route. Reactive protocols don't require the periodic updates of the routing table or cache which produces less routing overhead and bandwidth consumption. Some examples of reactive routing protocols are Dynamic Source Routing (DSR) [54], Ad Hoc On-Demand Distance Vector (AODV) [61], and Link Quality Source Routing (LQSR) [60].

3) Hybrid routing protocols. The hybrid routing protocols make use of the advantages of proactive and reactive routing protocols. They establish the efficient routes between all the nodes without the overhead of periodic updates or the path discovery process. One approach of the hybrid routing protocol is to use reactive protocols in the ad hoc portion of the network and proactive protocols in the backbone portion of the network. Examples of the hybrid
routing protocols include the Zone Routing Protocol (ZRP) [59], and the Hybrid Wireless Mesh Protocol (HWMP) [1].

1.4.1 Static vs. Dynamic Routing

In static routing, the routing information are entered and maintained manually by an administrator. In the case of topology changes, the routing table will not be updated automatically and will require the direct intervention of the administrator. The advantage of static routing is that it has less routing overhead. The disadvantages of static routing are the slow reaction to route changes and the difficult and impractical tasks of entering and maintaining routes in the routing table of all the nodes especially in large networks.

On the other hand, routing protocols with dynamic routing capability update the routing table periodically without the direct interaction by administrators. When a link breaks or a router goes down, the dynamic routing protocols automatically update the routing table in all nodes.

1.4.2 Routing Protocols

There are some important characteristics that must be considered when designing a new routing protocol for wireless ad hoc networks. The following are some of the most significant features:

1) Distributed. The routing protocol should be distributed among some or all of the nodes. The protocol should be decentralized even if the network contains some stationary nodes.

   This feature allows the network to be self-configured, self-organized, and self-healed.
2) Power conservation. In wireless ad hoc networks, mobile nodes have limited power sources. Therefore, a routing protocol must work with minimum control overhead. [63] and [64] discusses some aspects of the power consumption in ad hoc networks.

3) Multiple routes. Creating multiple routes to the same destination allows network to be self-healing. When a route becomes congested or has an increasing error rate, other routes can be utilized without the need to start a new route discovery process.

4) Secure. The nature of radio communications makes them susceptible to be intercepted by unauthorized users. This allows the intruder to perform some malicious activates in the network such as the man-in-the-attack, reply attack, manipulation of traffic, or denial of service. Therefore, the routing protocol should employ some security countermeasures such as encryption and authentication to protect the traffic traversing the network.

1.4.3 Routing Metrics

1) ETX. Expected Transmission Count is a metric which measures the quality of a path between two end nodes. When a packet is lost or dropped within an IEEE 802.11 network, the source retransmits the lost packet using Automatic Repeat Request (ARQ). Using a faulty link with high probability of errors will increase the number of retransmissions needed to send packets over that link. Therefore, the main objective of ETX is to find the path with the fewest retransmissions at the MAC layer. Due to the asymmetric links of the wireless network, ETX must probe both directions of the link because a successful transmission of packet requires the transmission of the data packet

11
in one direction and the transmission of the acknowledgment packet in the other
direction. Every node actively probes the links by broadcasting packets to the network. It compares the number of sent packets to the number of received packets over a period of time to determine ETX.

The disadvantage of ETX is that it doesn't take into consideration link capacity or node workload. Therefore, it may choose to transmit packets over links with low transmission rates. The delay in transmissions time causes other nodes to stop sending packets which leads to poor fairness when multiple nodes share a medium. Thus, ETX performs poorly in multi-radio networks but performs better than hop count in single-radio networks. ETX is used by some routing protocols such as OLSR and Babel [57].

2) ETT. The Expected Transmission Time (ETT) is an improvement of ETX by incorporating the different link capacities. When two links have the same error rates but different capacities, ETX seems to make bad decision because it doesn't consider the bandwidth. On the other hand, ETT uses the properties of ETX along with the link capacity when routing packets through the network which may increase the throughput and the fair medium sharing within the network.

However, ETT doesn't provide load balancing features because it doesn't measure the workload and can send packets through busy nodes and links. Moreover, ETT doesn't work properly in multi-radio networks because it does not take into consideration the intra-flow interference.

3) WCETT. The Weighted Cumulative Expected Transmission Time (WCETT) is an improvement of the Expected Transmission Time (ETT) routing metric. It was designed to work in multi-radio networks and therefore it considers channel diversity.
It favors shorter paths with high quality similar to ETT. Moreover, it prefers paths with high channel diversity and low intra-flow interference. It assigns the path with multiple links that use the same channel a high value.

The disadvantages of WCETT:

a. WCETT doesn’t consider the location of the links. If two links use the same channel but not in the same interference range, WCETT will dismiss this route even though interference is not possible.

b. WCETT is not isotonic and therefore very difficult to use in link state routing protocols.

4) MIC. The Metric of Interference and Channel Switching Cost (MIC) supports load-balancing routing. It is an isotonic metric which considers the intra-flow and inter-flow interference. MIC has two components which are Interference-aware Resource Usage (IRU) and Channel Switching Cost (CSC). IRU is concerned with inter-flow interference while CSC considers the level of intra-flow interference. MIC has two components which are Interference-aware Resource Usage (IRU) and Channel Switching Cost (CSC). IRU is concerned with inter-flow interference while CSC considers the level of intra-flow interference. The disadvantage of MIC is that it assumes that the neighboring nodes will continuously contend over the use of a single link even if that link wasn't active all the time.

5) iAware. The Interference Aware (iAware) metric is an improvement over the MIC. It proposes the use of a more accurate interference model by considering the amount of traffic generated by the nodes which are using the same channel within the same interference range.
1.4.4 Dynamic Source Routing (DSR)

Dynamic Source Routing (DSR) is a reactive routing protocol. It creates a path from the source to the destination upon the request of the source [53]. Also, it is a source routing protocol. Each packet contains complete information about every node along the selected path including the IP address of each node. Hop-count. It was designed to reduce the overhead on the network and to be a self-organizing and self-configuring protocol for WMN [54]. DSR has two phases: Route Discovery (RD), Route maintenance (RM).

1. Route discovery: when a node attempts to send data to other nodes, it takes the following steps to establish an end-to-end path:
   a. The source floods the network with Route Request packets (RREQ). RREQ contains the source IP address and the destination IP address. Figure 5 shows the RREQ header format.
   b. If the RREQ was received by an intermediate node that is not the destination, it appends its information to the RREQ and broadcasts it to all neighboring nodes.
   c. When the intended destination node receives the RREQ, it adds the source route to a Route Replay packet (RREP) and send it back to the source node through the same route of the RREQ. Every node stores multiple route entries for each destination node in the network in a routing cache.

2. Route maintenance: it happens when a link between source and destination is broken or when a mobile node changes its location. A node can maintain a route by taking the following steps:
a. Each node forwards packets to the next node in the route and acknowledges it to the previous node. If the link is broken, the acknowledgement will not be sent back to the source.

b. When an intermediate node discovers a broken link, it sends a Route Error packet (RERR) to the source.

c. The source node will stop sending packets and check its route cache for alternative paths.

Figure 1.5 The header format of the RREQ packet in DSR

The advantages of DSR:

1) Route discovery and route maintenance allows DSR to be self-configuring and self-organizing which is one of the requirements of ad hoc networks.

2) DSR doesn't require periodic updates regarding neighbors and link state which reduces the routing overhead.

3) DSR can store multiple routes to the same destination in the route cache. If the link was inactive, it checks the route cache for alternative routes and if found, no need for a new route discovery.
4) DSR can deliver packets through different types of networks because it is a layer 3 protocol. It supports nodes with multiple network interfaces of different types forming an ad hoc network [53].

5) Intermediate nodes can save partial route information to use it to answer next route requests faster. However, this feature only works if the topology has limited changes.

6) Intermediate nodes can use route information embedded in packets to update its route cache.

A disadvantage of DSR is that it performs poorly in networks with highly dynamic topologies and unreliable links.

1.4.5 Adhoc On-Demand Distance Vector (AODV)

Ad-hoc On-Demand Distance Vector (AODV) is a distance vector routing protocol for wireless ad-hoc networks. It is similar to DSR as it is a reactive routing protocol [56]. AODV is a table-driven routing protocol unlike DSR which uses source routing. Under AODV, each node keeps only one record for each destination node in the network. AODV is capable of performing both unicasting and multicasting. As the name implies, it works on-demand meaning it only creates a route between nodes when requested by the source. A sequence number is added the Route Request packet (RREQ) to ensure the freshness of route. AODV is loop free and highly scalable.

Route discovery in AODV takes the following steps:

1) The source flood the network with a RREQ for a route to the intended destination. Figure 6 shows the RREQ header format.
2) If an intermediate node receives the RREQ and has a fresh route to the destination with a destination sequence number that is greater than or equal to the one in the RREQ, it sends a Route Reply packet (RREP) to the source which contains the route information.

3) If the intermediate doesn't have a fresh route to the destination, it will increment hop count by one and broadcast the request to all neighbors. To prevent loops, nodes keep the source IP address and the broadcast ID.

4) If a node has received a duplicate request due to broadcasting, that request will be discarded.

5) When the RREP is sent back to the source node, each node on the route will establish a pointer to the destination node.

6) If the source node received another RREP to the original RREQ with an equal or greater destination sequence number but with lower hop count, it will update its routing table accordingly.

7) If the route was not active meaning the source is not sending packets to the destination for a certain period of time, the path information will be timed out and deleted from the routing tables of the intermediate nodes. In this case, the source will need to initiate the discovery process by broadcasting a new RREQ.

8) If a link goes down within the route between the source and destination, the upstream node will send a Route Error message (RERR) to the source.
1.4.6 Link Quality Source Routing (LQSR)

Link Quality Source Routing (LQSR) is reactive routing protocol [60]. It was developed by Microsoft based on the DSR routing protocol. It is a 2.5-layer protocol between the network layer and the data link layer whereas DSR is network layer protocol.

The Advantages of LQSR:

1) LQSR is transparent to other layers in the OSI model and therefore no modifications needed on the other layers.

2) It inherent all the functionalities of the DSR routing protocol such as route discovery and route maintenance.

3) However, it employs multiple link quality metrics such as hop count, EXT, Per-hop RTT, and Packet Pair.

4) It uses Weighted Cumulative Expected Transmission Time (WCETT) as a metric. Each link to the destination is assigned with a relative weight. LSQR chooses the link with the highest weight.
5) It continuously monitors link quality and calculates the overall cost of all active links and chooses the lowest cost link.

1.4.7 Optimized Link State Routing (OLSR)

Optimized Link State Routing (OLSR) is a proactive routing protocol for wireless ad-hoc networks where routes are determined ahead and available immediately when needed [58]. OLSR is a table-driven routing protocol. It is an optimized version of a pure link state routing protocol. It continuously keeps fresh routes to all the destinations in the network. However, only the nodes assigned as Multipoint Relays (MPR) can broadcast routing table updates to the network. This feature is useful in reducing the number of broadcasted updates and therefore conserving the bandwidth. Each node maintains a list of MPR nodes. Figure 7 shows MPR nodes in the network topology when using OLSR.

There are two types of control messages in OLSR:

1) Hello message: It is used to identify two-hop neighbors.

2) Topology Control message (TC): It is used to broadcast routing table information to the other nodes. Nodes use the information included in the Topology Control message to update their routing tables. Figure 8 shows the packet format of OLSR.
1.4.8 Destination Sequenced Distance Vector (DSDV)

The Destination Sequenced Distance Vector protocol (DSDV) [55] is a table-driven routing protocol for wireless ad-hoc networks. It is a distance vector protocol based on the
Bellman-Ford algorithm. It is a proactive routing protocol that maintains fresh routes to each node in the network by periodically updating the routing table.

Each node sends routing table updates to all neighboring nodes within a certain period of time or upon significant topology changes. Each record in the routing table contains all available destinations, hops number to the destination, and a sequence number. The sequence number is assigned by the destination and it is used to prevent routing loops. If there are more than one entry to the same destination, the router chooses the route with the highest sequence number. If two or more routes have the same sequence number, the router chooses the shortest route.

The routing updates can be sent using one of the following methods:

1. Full dump: send the entire routing table to all neighboring nodes.
2. Incremental update: send only recently updated entries.
1.5 Security in WMN

IEEE 802.11s implements the Mesh Security Association (MSA) to extend the Robust Security Network Association (RSNA) of 802.11i. The 802.11s standard defines two types of key holders: Mesh Authenticator (MA) and Mesh Key Distributor (MKD). A Mesh Point (MP) can assume either one or both functions. A mesh point refers to any member of the mesh network including mesh clients (STA), mesh routers, servers, and mesh gateways. The 802.11s standard provides secure associations between all the mesh points using the MSA.

802.11s introduces the concept of mesh key hierarchy to facilitate a secure MSA link between the mesh points. It inherits the security functions of 802.11i with some extensions [20]. For example, it uses 802.1X for the initial authentication process. The mesh key hierarchy consists of two branches:

1. Secure link for the initial authentication.
2. Key generation and distribution.

On the first phase of the secure link branch, the MP supplicant and the MKD establish the mesh key hierarchy during the initial authentication. A Pairwise Master Key (PMK-MKD) is mutually derived on both the MP supplicant and the MKD using a Pre-Shared Key (PSK) or a Master Session Key (MSK). MSK is the preferred solution when 802.1X authentication is enabled [19].

On the second phase of the secure link branch, the Pairwise Master Key (PMK-MA) between the supplicant MP and the MA is generated mutually between the supplicant and the MKD. The PMK-MA is the delivered by the MKD to the MA using the MSA. The MA initiates a 4-way handshake with the supplicant MP and requests the
Group Transient Key (GTK), which is a shared key between all the supplicant MPs that are connected to the same MA.

On the key distribution branch, a Key Distribution Key (KDK) using a PSK or a MSK. Finally, a Pairwise Transient Key (PTK) is derived for the key distribution function. Figure 1.9 demonstrates the full authentication and key generation process in WMN.

Figure 1.9 Authentication and key generation in 802.11s
1.6 Research Motivation

1.6.1 Lack of Fast Handoff Mechanisms for WMNs

There are many challenges facing the wide deployment of WMN such as routing, security, QoS, and privacy. This dissertation is focused on the different aspects of the handoff authentication process in WMN. The current 802.11s standard does not specify any handoff authentication mechanisms for WMN. Instead, it relies on the IEEE 802.11i standard [2], the security standard for wireless networks, and its subsidiary IEEE 802.1X [3] to provide the security services and mechanisms needed. Therefore, WMN shares similar performance and security issues facing other traditional wireless networks.

The absence of a fast handoff mechanism for WMN raises the need to design a new handoff protocol to satisfy the unique specifications of this new technology. Authentication protocols designed originally for the traditional wireless LAN have proved to be slow and inefficient when used in WMN. For example, when a client moves between access points, it must initiate the full authentication process with the Authentication Server (AS) using one of the Extensible Authentication Protocol (EAP) methods such as EAP-TLS, which requires the exchange of nine authentication messages [11]. In such case, the authentication messages must traverse the network through multiple hops between the client and the AS in order to complete the authentication process. As a result, the handoff latency may increase (up to several seconds) relatively to the network diameter [5]. Therefore, service degradation or interruption becomes inevitable as the maximum delay required for some real-time applications cannot exceed 50 ms [38]. The International Telecommunication Union (ITU-T) recommends a 150 ms maximum one-way delay for VoIP applications [9]. Moreover, the Internet Engineering Task Force (IETF)
recommends a 100 ms maximum delay for first person online games, 200 ms for remote desktop applications, and 5 seconds for instant messaging [68].

We have conducted an experiment based on an actual setup of a multi-hop wireless network to measure the delay that occurs when 802.1X/EAP authentication methods are utilized. The results from this experiment shows that the delay when using EAP-TLS is about 3.5 seconds. The delay occurs when using EAP-PEAP is about 800 ms. Notice that in both cases, the delay far exceeds the maximum latency requirements for most of the aforementioned application. Hence, designing a fast and efficient handoff authentication protocol for WMN to minimize the handoff delay and improve user experience is a practical research area.

For the purpose of this dissertation, we conducted preliminary experiments to evaluate the cost of running cryptographic primitives on mass produced mobile devices. The objective of these experiments is to determine which cryptographic methods can be a suitable choice for the handoff authentication protocol in WMN. The new protocols must provide a high level of security while maintaining seamless roaming and operational continuity. A performance evaluation framework has been developed to analyze the performance of different available handoff authentication protocols. Performance analysis tools for Android and iOS has been developed based on the specifications of the proposed framework.
1.6.2 The Increased Threats Against Wireless Networks

Wireless networks are vulnerable to many attacks due to the open nature of the shared medium and the lack of physical access controls. Some of the most anticipated attacks against wireless networks include:

a. **Passive eavesdropping:** In wireless networks, it is an easy task for an adversary to sniff the traffic due to the shared nature of the wireless medium. Even if the messages were encrypted, the attacker still can gather and store significant information such as message types and header attributes. The stored data and some known plaintext can then be used to launch offline attacks to retrieve the encryption key. It is almost impossible to detect and protect against these types of attacks.

b. **Message injection:** An attacker can generate legitimate 802.11 packets using a laptop with wireless capability and tools available on the internet. The attacker can control the content of the message, the header fields, and the transmutation methods. Even if authentication was required, the attacker still can generate a valid packet by exploiting the weaknesses of the authentication protocols that use data integrity protocols such as MD5 by performing intense brute force and dictionary attacks [8]. In addition, the attacker can intercept and replay packets in networks that do not provide protection against replay attacks.

c. **Message deletion:** An attacker deletes a message by forcing the receiver to drop that message. The most common way to perform this attack successfully is by causing a Cyclic Redundancy Checksum (CRC) error at the receiver [8]. The receiver can’t distinguish if the error was caused by noise or by a deliberate attack. The receiver will drop the packet and request a retransmission. In a more advanced variation of
this attack, the attacker can use two antennas. The first antenna is used to force a packet drop at the receiver side and the second is used to sniff the packet from the shared medium. The attacker can then decide whether or not to send the packet to the receiver.

d. *Rouge access points:* Each packet in the network contains the MAC addresses of both the sender and the receiver in plaintext. An attacker can exploit that by sniffing the traffic and learn the MAC addresses. The attacker then can spoof its MAC address to appear as a valid client or access point (AP). The attacker can perform the normal tasks of a legitimate access point by using available tools such as HostAP [8].

e. *Session hijacking:* An attacker can hijack a session between two wireless entities after they successfully authenticate each other. One way to perform this attack is by causing a disconnection on one side of that connection and then masquerade as the disconnected side. This attack is very difficult to mount and can be averted using strong data confidentiality and integrity protocols.

f. *Man-in-the-middle:* An attacker can hijack the session between a client and an access point by disconnecting the connection between them. The attacker then can proceed to masquerade as the client and authenticate itself with the access point. After that, it can masquerade as the access point and authenticate itself with the client. If the attack was successful, the attacker can forward the packets between the client and the access point without raising suspicion. There are some attacks to help mounting the man-in-the-middle attack such as MAC spoofing and ARP cache poisoning.
1.7 Dissertation Contributions

In this dissertation, two handoff authentication protocol have been designed and implemented to address the issues stemming from the utilization of resource-craving and delay-prone protocols such as EAP-TLS and EAP-PEAP. Unlike current authentication protocols, our protocols, UFAP and GAP, do not require the pre-existence of a centralized key management structure, such as a Public Key Infrastructure (PKI). Centralized authentication systems, designed for traditional wired and wireless LAN networks, are not a favorable choice for a distributed network such as the case of WMN. In addition, our protocols provide better performance in networks with dynamic topology, high mobility, and constant changes in QoS provisioning which may affect the overall performance of the network. For example, the Ping-Pong effect, which happens as a result of sudden changes in link quality, may lead to significant delays due to the high frequency of handoffs.

Our first protocol, UFAP, is an ultra-fast and highly efficient token-based handoff authentication protocol for WMN. This protocol reduces the handoff delay by delegating the authentication process to the access points at the network edges rather than the centralized authentication servers. It eliminates the communication delay incurred due to the multi-hop exchange of authentication messages. Furthermore, it increases network efficiency by reducing the number of messages exchanged during the handoff authentication.

The second protocol, GAP, is a fast and secure ticket-based handoff authentication protocol. It is a variation of the first protocol designed to address some conditions such as
constant topology changes, connectivity problems, and erratic movement patterns. This protocol reduces the handoff delay by utilizing a global handoff ticket which can be recognized and accessed by any legitimate access point in the network.

Our third contribution is the design and development of the Handoff Authentication Latency Assessment framework (HALA). This framework is designed to evaluate the performance of current and proposed security protocols on mobile devices. In general, security protocols consist of a set of cryptographic primitives such as hash functions, symmetric, and asymmetric encryption. The framework provides an easy method to build and measure the performance of the proposed protocols. The results are presented in the form of raw, a comprehensive analysis of the final results, and a comparison with previously established baselines. In addition, this dissertation provides detailed results from experiments designed to measure the execution times of cryptographic primitives on recent Android and iOS mobile devices. These results can help researchers determine the possibility of using some cryptographic methods in delay-sensitive applications such as real-time applications and applications with certain deadlines.
1.8 Dissertation Outline

This dissertation is organized as follows. Chapter 2 provides general overview of current and proposed methods and protocols used for authentication in WMN. It presents the specifications of different symmetric and asymmetric key cryptographic methods, such as AES, RSA, and ECC, commonly used in security schemes. This section aids our research efforts to distinguish which of these methods can be employed in the design and implementation of new handoff protocols. Moreover, it examines current security standards and implementations such as the IEEE 802.11i security standard, the IEEE 802.1X port-based authentication standard, and IEEE 802.11r standard for Fast BSS Transition. It discusses the Extensible Authentication Protocol (EAP) which provides several authentication methods used in WMN and defined in the 802.1X standard, such as EAP-TLS, EAP-TTLS, and EAP-PEAP. We also provide detailed reviews of proposed handoff authentication protocols to establish a baseline for the evaluation of our new protocols.

In Chapter 3, we provide an overview of our first contribution, the Handoff Authentication Latency Assessment (HALA) framework. This framework can be used to analyze the performance of proposed security schemes by measuring the execution time of each component of these protocols. We discuss the details of the three-layer design and implementation of the framework and how it can be utilized to compare the performance of a security protocol with other benchmarks.
Moreover, we provide the specification of the security systems in the most popular mobile platforms, Android and iOS. Performance analysis tools are developed for both platforms based on the HALA framework to measure the execution time for different cryptographic methods. The experiments have been conducted on several Android and iOS mobile devices and the results are discussed.

In Chapter 4, we introduce our first handoff authentication protocol, the Ultra-Fast Authentication Protocol (UFAP). We discuss the design principles and requirements which have been taken into consideration during the initial design phase. We provide the details of the new design and the handoff process for this protocol. A small testbed is established in an actual setup to measure the performance of the UFAP protocol in comparison with other authentication protocols including EAP-TLS and EAP-PEAP. We discuss the advantages of this protocol from the performance perspective including the short delays, efficiency, and security overhead. Finally, we discuss the security countermeasure embedded in the design of this protocol to provide protection against well-known attacks.

In Chapter 5, we present the second handoff protocol, the Global Authentication Protocol (GAP). We discuss the design aspects of the GAP protocol and the details of the handoff process. We compare the performance of GAP when used in the infrastructure WMN and in the hybrid WMN with the first protocol, UFAP. Moreover, we compare the performance of GAP with current authentication protocols including EAP-TLS and EAP-PEAP.
In Chapter 6, we provide a conclusion and an insight in the future research directions. We discuss the advantages of our handoff authentication protocols, UFAP and GAP, from the performance and security perspectives in comparison with other authentication protocols. We explore some future research ideas to improve the work discussed throughout this dissertation. For example, we suggest the integration of our protocols with other technologies such as MobileIP (MIP) to provide fast and secure inter-network transition. Another potential research involves the design of new secure routing protocols and secure delivery methods. Finally, we discuss some of the issues arose during the different production phases of this dissertation and some of the solutions implemented to resolve these issues.
CHAPTER
II. RELATED WORK

2.1 Cryptographic Methods Used in Authentication

2.1.1 Public Key Cryptography

2.1.1.1 Diffie-Hellman

The concept of asymmetric encryption was introduced by Whitefield Diffie and Martin Hellman in 1976 [39]. Diffie and Hellman proposed a two-key system, a public key and a private key, where either key can be used for the encryption and decryption of the data. A client can share its public key with others while keeping the private key secret. It takes advantage of the unique properties of the arithmetic modular to make it infeasible to deduce the private key by the mere knowledge of the public key. The following is the process of generating the keys in Diffie-Hellman (DH) by two parties, Alice and Bob:

1. Alice and Bob agree on two large prime numbers \( P \) and \( N \) which would be publicly shared.
2. Alice choose a large number \( A \), which must be relatively prime to \( N \), as a secret component known only to Alice.
3. Alice calculates \( C = N^A \pmod{P} \) and sends \( C \) to Bob.
4. Bob choose a large number \( B \), which must be relatively prime to \( N \), as a secret component known only to Bob.
5. Bob calculates \( D = N^B \pmod{P} \) and sends \( D \) to Alice.
6. Alice calculates \( E_1 = D^A \pmod{P} \) and Bob calculates \( E_2 = C^B \pmod{P} \).

Notice that \( E_1 \) and \( E_2 \) are equal and therefore, Alice and Bob managed to reach a key agreement despite the fact that neither of them knows the secret component of the other.
party. Even though, P, N, C, and D are made public by Alice and Bob, a third party, Eve, can’t generate \( E_1 \) without knowing the secret components A and B. The following is to prove that notion:

\[
E_1 = D^A \pmod{P} \Rightarrow \text{This was computed by Alice} \\
= (N^B)^A \pmod{P} \\
= N^{BA} \pmod{P}
\]

\[
E_2 = C^B \pmod{P} \Rightarrow \text{This was computed by Alice} \\
= (N^{-1})^B \pmod{P} \\
= N^{AB} \pmod{P}
\]

Therefore, \( E_1 \) and \( E_2 \) are equal and both Alice and Bob share the same key.

The Diffie-Hellman algorithm provides a viable solution to some of the shortcomings of the symmetric cryptography include key distribution and scalability. In symmetric cryptography, one secret key is shared between two parties and used to encrypt and decrypt data. Although much simpler than public key cryptography, the key must be delivered in a secured manner which is a major issue especially with the lack of secure communication channels. Scalability is another disadvantage of symmetric cryptography as the number of shared secret keys grows rapidly as the members’ number increases in the network. The number of shared keys needed within the network can be determined using

\[
a = \frac{N(N-1)}{2}, \text{ where } N \text{ is the total number of users.}
\]

2.1.1.2 RSA

Another widely used public key cryptosystem is RSA. It was developed at MIT in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman, hence the abbreviation (RSA) [40]. Similar to DH, RSA benefits from the properties of modular arithmetic and prime
numbers. RSA is considered highly secure due to difficulty of factoring a large number into two large prime numbers.

The following are the steps to generate the public key and private key in RSA:

1. Alice picks two large prime numbers P and Q which will be kept as a secret.
2. Alice calculates the modulus N, where N = P * Q.
3. Alice calculates the totient φ(N) = (P-1) (Q-1).
4. Alice picks the encryption exponent e, where 1 < e < φ(N) and both e and φ(N) are relatively prime (gcd (e, φ(N)) = 1).
5. Alice computes the decryption exponent d, where $e \times d \mod \phi(N) = 1$, using the extended Euclidean algorithm.

In this case, the public key consists of the modulus N and the encryption exponent e (e, n) and it is published by Alice. The private key consists of the modulus N and the decryption exponent d (d, n) and only Alice has access to it. To encrypt a message M using Alice’s public key, Bob computes $C = M^e \mod N$ and sends it to Alice. Alice decrypts the message C by computing $M = C^d \mod N$. RSA is a block cipher and the message size must be less than the modulus N.

Public key schemes were developed to solve the key distribution problem. Symmetric key cryptography requires the distribution of the secret keys via secure channels which could be a difficult and inefficient task. In RSA, the public key can be made available to other users through some publication methods such as key depository systems (Key Servers), or digital certificates produced by a Certificate Authority (CA). Examples of
available key servers, which allow users to search for specific public keys, are keyserver.pgp.com, pgp.mit.edu, and keyserver.ubuntu.com.

A major setback of using public key and should be considered when designing an authentication scheme for WMN is performance. RSA uses modular arithmetic to perform encryption and decryption functions which requires heavy computation. A performance analysis study was conducted on RSA authentication using 1024-bit and 2048-bit key sizes on a handheld [41]. The study shows that the average time required for a full handshake is 1.14 seconds using 1024-bit keys and 2.06 seconds using 2048-bit keys. The study also shows that the energy consumption increases significantly when using RSA for authentication.
2.1.1.3 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a public key scheme based on the discrete logarithm problem of elliptic curves. The points of an elliptic curve are members of the same finite field. The concept of using elliptic curves in cryptography was introduced independently by Victor Miller [42] in 1985 and Neil Koblitz [43] in 1987. An elliptic curve $E$ over a finite field $K$ is defined using the following equation:

$$E: y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6,$$

where $a_1, a_2, a_3, a_4,$ and $a_6 \in K$

The following are the steps to generate the public key and the private key using ECC:

1. Alice and Bob agree on some public parameters including the finite field, the elliptic curve, and a base point $G$ on the curve, and the order $n$ of the finite field.

2. Alice picks a random number $d_A$ between $\{1, \ldots, n-1\}$ as the private key and computes and publish the public key $H_A = G d_A$.

3. Bob picks a random number $d_B$ between $\{1, \ldots, n-1\}$ as the private key and computes and publish the public key $H_B = G d_B$.

4. Alice computes the secret key $S = d_A H_B$

5. Bob computes the secret key $S = d_B H_A$

Both Alice and Bob now share the same secret key because:

$$S = d_A H_B = d_A (G d_B) = d_B (G d_A) = d_B H_A$$

The security level of ECC relies on the difficult problem of discrete logarithm. An attacker cannot deduce the secret key $S$ just based on the knowledge of the public keys $H_A$ and $H_B$. 
ECC offers an overall faster performance and requires less memory than RSA. Moreover, ECC provides the same security level of RSA using a smaller key size. For example, 160-bit and 224-bit ECC keys provide equivalent security levels to 1024-bit and 2048-bit RSA keys respectively [44]. A performance analysis study conducted on ECC and RSA using different key sizes shows that ECC has faster performance in key generation and encryption. On the other hand, RSA has faster performance in decryption and signature verification [45].

2.1.2 Symmetric Key Cryptography

Symmetric key encryption is a form of cryptography in which the same secret key is used for both the encryption and decryption of data. In this scheme, the secret key is generated by one party and then transmitted to the second party over a secure communication channel. Some examples of symmetric key cryptosystems include DES, 3DES, AES, and Blowfish. There are two parts of a symmetric cryptosystem: the encryption algorithm and the encryption key.

![Symmetric key encryption and decryption](image)

Figure 2.1 Symmetric key encryption and decryption

The encryption algorithm, in most cases, is published and should be assumed to be the target of extensive cryptanalysis. On the other hand, the encryption key must be kept secret. It is an essential factor in determining the security level of a symmetric
encryption protocol. Choosing the right key length is an important countermeasure in order to deter cryptanalysis attacks aimed to exploit the encryption algorithm to deduce the encryption key. Some of the well-known attacks against symmetric key cryptosystems are brute-force attack, related-key attack [28], and the side-channel attack [29].

Table 2.1 A comparison between DES, 3DES, AES, and Blowfish

<table>
<thead>
<tr>
<th></th>
<th>DES</th>
<th>3DES</th>
<th>AES</th>
<th>Blowfish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Size</td>
<td>64</td>
<td>64</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>Key Size</td>
<td>56</td>
<td>168</td>
<td>128, 192, 256</td>
<td>32-448</td>
</tr>
<tr>
<td># of Rounds</td>
<td>16</td>
<td>48</td>
<td>9, 11, 13</td>
<td>16</td>
</tr>
</tbody>
</table>

There are several advantages for using symmetric key encryption over public key encryption. First, symmetric key encryption algorithms are generally faster than public key encryption algorithms. The encryption operations used in symmetric encryption algorithms are substitution, transposition, and multiplication. On the other hand, public key encryption algorithms rely on complex mathematical operations such as modular arithmetic and discrete logarithms which require heavy computations. Second, symmetric encryption algorithms provide equivalent security levels to public key encryption algorithms using small encryption keys.

The main disadvantage of symmetric key encryption is scalability. The number of keys required to have secured communications rises significantly as the number of users increases in the network. The number of shared keys is \( N = \frac{N(N-1)}{2} \), where \( N \) is the number of users. Moreover, symmetric key encryption cannot be used to generate digital signatures.
Table 2.2 Equivalent security comparison by key size

<table>
<thead>
<tr>
<th>AES</th>
<th>RSA</th>
<th>ECC</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>1024</td>
<td>163</td>
</tr>
<tr>
<td>N/A</td>
<td>2240</td>
<td>233</td>
</tr>
<tr>
<td>128</td>
<td>3072</td>
<td>283</td>
</tr>
<tr>
<td>192</td>
<td>7680</td>
<td>409</td>
</tr>
<tr>
<td>256</td>
<td>15360</td>
<td>571</td>
</tr>
</tbody>
</table>
2.2 Current Wireless Security Schemes

2.2.1 Wired Equivalent Privacy

Wired Equivalent Privacy (WEP) is an encryption protocol designed by a group of IEEE members as a part of the 802.11 standard for wireless networks [9]. As the name implies, WEP was supposed to provide a high level of security for wireless networks that implement the 802.11 standard similar to that of wired networks. Each packet sent over the shared wireless medium would be encrypted to protect the confidentiality of the data.

In WEP, each packet is encrypted individually using the RC4 cipher stream with a 64-bit key. The encryption key is a combination of a 24-bit Initialization Vector (IV) and 40-bit WEP key. Figure 2.2 shows the content of a WEP packet.

<table>
<thead>
<tr>
<th>IV</th>
<th>Encrypted Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 bits</td>
<td>Data</td>
</tr>
<tr>
<td>Padding 6</td>
<td>ICV 32 bits</td>
</tr>
<tr>
<td>2 bits</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.2 WEP packet format

The following steps describe the encryption process using WEP [10]:

1. The sender chooses the IV which can be changed for each packet. Therefore, each packet will be encrypted using a different combination of IV and WEP key. This feature was added to protect WEP against offline cipher attacks. If the attacker managed to decrypt one packet, the key used to encrypt that particular packet will not be valid to decrypt other packets.

2. The packet is encrypted by using a bitwise exclusive OR (XOR) on the plaintext and the combination of IV and the WEP key.

3. A packet including the resulting ciphertext and the IV appended in plaintext is sent to the receiver.
4. The receiver, which has a copy of the shared WEP key, uses the WEP key and the appended IV to decrypt the packet.

Over the years, many vulnerabilities in both the authentication and encryption processes of WEP were identified by researchers deeming it insecure and eventually obsolete. In 2001, a paper published by Borisov, Goldberg, and Wagner noted the design blunders of WEP and concluded that the use of ICV doesn’t protect against malicious attacks [11].

Some of the well-known vulnerabilities of WEP include:

a. Using a shared key: It is difficult to change the shared WEP key as it requires the key to be manually modified in all wireless clients because WEP doesn’t provide key management. As a result, WEP keys tend to have long life as they are rarely changed and therefore vulnerable to some cipher attacks such as brute force and dictionary attacks.

b. Using RC4: RC4 has been scrutinized by researcher for generating weak keys [10]. In particular, attackers can discover more relations between the encryption key and the output of the encryption process.

c. Small key size: The most noted weakness in WEP is the key size. The key size for WEP specified in the 802.11 standard is 40-bit. Although, some vendors apply 104-bit or 232-bit keys [10].

d. Reusing IV’s: The small-sized 24-bit IV has only 16,777,216 different values. This number can be depleted in a short period of time leading to duplicate encryption keys [10]. Attackers can use the encryption keys with the same IV do retrieve the shared
WEP key. The IV and the shared WEP key can be used to decrypt legitimate packets or send forged packets.

e. No key management: Key management is a daunting task when it comes to WEP. The 802.11 standard for WEP doesn’t specify any key management protocol. A traditional wireless network has one WEP key shared among wireless clients and access points. Therefore, a change in the password would require manual configuration by network administrators on all machines.

With the noted vulnerabilities of WEP, it wasn’t long before the involved parties acknowledge that WEP doesn’t provide adequate security for wireless networks. There are many alternative solutions to WEP proposed by several vendors including:

a. WEP+: A 104-bit key to extend the 40-bit WEP key. It was proposed in 1998 by Lucent. The key feature of WEP+ is that it avoids using weak keys. However, the same weaknesses of the original WEP can be exploited in WEP+ [9].

b. WEP2: An enhancement to WEP proposed by IEEE as a part of the 802.11i standard although dropped later. It increases the combination of IV and WEP key from 64-bit to 128-bit. The goal for this enhancement is to decrease the probability of duplicate IVs [10].

c. Dynamic WEP: It is a vendor-specific solution provided by several vendors such as Cisco, 3Com, and Microsoft. The access points generate and broadcast short-lived keys over preconfigured time intervals. The keys are established using the shared WEP key [9].
2.2.2 Wi-Fi Protected Access

Wi-Fi Protected Access (WPA) is a security standard proposed by the Wireless Fidelity (Wi-Fi) Alliance, a non-profit international association which certifies the interoperability of wireless network devices based on the IEEE 802.11 standard. The new WPA standard is a subset of the IEEE 802.11i finalized by IEEE. It is also known as Robust Security Network (RSN) [11]. It was designed to be compatible with the Wi-Fi products with WEP enabled to promote an easy transition from WEP to WPA.

Unlike WEP, WPA separates the encryption process and the user authentication process. WPA implements the Temporal Key Integrity Protocol (TKIP) for data protection. TKIP uses RC4 stream cipher as the encryption protocol and all network clients and access points share a 128-bit secret key called Temporal Key (TK). TKIP also has the option of using the Advanced Encryption Standard (AES) as the encryption protocol instead of RC4. Furthermore, TKIP uses a 48-bit Initialization Vector (IV) similar to WEP. However, unlike WEP, TKIP generates a Per-Packet KEY (PPK) using the TK key. The unique PPK is then used as the encryption key for the RC4 stream [10].

Another mechanism in TKIP which wasn’t featured in WEP is the message integrity. TKIP uses a 64-bit Message Integrity Code (MIC) which is a code generated using an algorithm called Michael [9]. MIC is used to detect both transfer errors and malicious modifications. In addition, an Integrity Check Value (ICV) is used to detect errors in both the data and the MIC. Figure 2.3 shows the TKIP packet format and content.

<table>
<thead>
<tr>
<th>IV</th>
<th>Extended IV</th>
<th>Encrypted Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>IV 16 bits</td>
<td>RSV 4 bits</td>
<td>Data</td>
</tr>
<tr>
<td>Ext IV 1 bit</td>
<td>Key ID 2 bits</td>
<td>MIC 64 bits</td>
</tr>
<tr>
<td>IV 32 bits</td>
<td></td>
<td>ICV CRC-32</td>
</tr>
</tbody>
</table>

Figure 2.3 TKIP packet format
2.2.3 IEEE 802.11i Security Standard

IEEE 802.11i [2] is a standard designed by IEEE to provide enhanced security for 802.11 wireless networks at the Medium Access Control (MAC) layer. The 802.11i standard introduced the concept of Robust Security Network (RSN) which provides data protection, mutual authentication, and key management. RSN allows the implementation of TKIP or AES as the data confidentiality protocol. By default, 802.11i utilizes AES as the encryption protocol along with the Counter-mode Cipher Block Chaining Message Authentication Code (CBC-MAC) Protocol (CCMP) [8].

IEEE 802.1X, a sub-entity of the 802.11i standard, is a port-based access control which supports multiple authentication services. 802.1X provides a major enhancement to 802.11i over the Wired Equivalent Privacy (WEP) by adding mutual authentication and key management using the Extensible Authentication Protocol (EAP). EAP utilizes a secure public-key encryption system. Only authorized users can gain access to the network [10].
2.2.4 802.1X/EAP Port-Based Authentication

IEEE 802.1X [3] is an enhancement to the IEEE 802.11i standard by providing mutual authentication, authorization, and key management. It is defined in the released IEEE 802.1X document as,

"Port-based network access control makes use of the physical access characteristics of IEEE 802 LAN infrastructures in order to provide a means of authenticating and authorizing devices attached to a LAN port that has point-to-point connection characteristics, and of preventing access to that port in cases which the authentication and authorization fails. A port in this context is a single point of attachment to the LAN infrastructure."[3].

It was initially design for the wired networks but was later optimized to work in 802.11 wireless networks. There are three major elements defined in the 802.1X standard [9]:

a. Supplicant: a mobile wireless client seeks to be authenticated and associated with network.

b. Authentication server: the entity which performs the actual authentication process. An example of the authentication server is RADIUS [14].

c. Authenticator: the device which serves as the intermediary between the supplicant and the authentication server. It performs port-based access control by allowing only authenticated client to access the network using uncontrolled ports. Typically, the access point contains the Authenticator.

In 802.1X, the Authenticator manages several logical ports which are attached to the same physical port. There are two types of logical ports which are controlled and uncontrolled ports. An unauthenticated supplicant can only have access to the network via uncontrolled and only EAP traffic to the authentication server is allowed. Once authenticated, the supplicant can gain access to other network resources via controlled ports [14].
Figure 2.4 shows the mutual authentication steps of the 802.1X [9]:

1. The Supplicant sends an authentication request to the Authenticator.

2. The Authenticator grants the Supplicant access to the network via an uncontrolled port. Only EAP traffic is allowed at this point.

3. The Authenticator sends an identity request to the Supplicant.

4. The Supplicant sends its identity to the Authenticator.

5. The Authenticator relays the received identity to the Authentication Server via an uncontrolled port.

6. The Authentication Server performs the authentication process and sends an ACCEPT message to the Authenticator if the Supplicant was authenticated successfully.

7. The Authenticator transfer the supplicant to a controlled port.

8. The Supplicant requests the identity of the Authentication Server.


10. The Supplicant authenticates the Authentication Server and starts accessing the network resources.
Figure 2.4 802.1X authentication process
2.2.5 The Extensible Authentication Protocol

The Extensible Authentication Protocol (EAP) is the transport protocol used by 802.1X to transfer the data between the involved parties during the mutual authentication process. EAP is a flexible framework which allows the use of different authentication methods such as passwords, certificates, Kerberos, and smart cards. EAP works over data link layer protocols such as PPP and IEEE 802 and provides support for retransmission.

Some of the authentication methods used in EAP:

1. EAP-LEAP: Lightweight EAP is a propriety authentication protocol developed by Cisco. In LEAP, user credentials (username and password) are sent to a Remote Authentication Dial-In User Service (RADIUS) server for authentication [14].

According to Cisco,

"Cisco LEAP is a mutual authentication algorithm that supports dynamic derivation of session keys. With Cisco Leap, mutual authentication relies on a shared secret, the user’s password, which is known by the client and the network and is used to respond to challenges between the user and the RADIUS server" [15].

LEAP is considered insecure due to the following reasons:

a. Attackers can exploit the shared wireless medium by observing the traffic and intercepting the challenge and response exchanged between the Supplicant and the Authentication Server.

b. The username is sent in clear text.

c. The challenge is encrypted using DES which has been proven to be the subject of many successful cryptanalysis attacks. In 1997, the Electronic Frontier Foundation (EFF) produced a special purpose machine which was used to retrieve a DES encryption key in less than three days [16].
2. EAP-PEAP: Protected EAP is an authentication protocol developed by Cisco, Microsoft, and RSA to overcome the weaknesses of other EAP authentication methods. It allows the use of many security implementations such as MS-CHAPv2 and TLS-tunnel to secure the connection between the Supplicant and the Authentication Server. It requires the use of PKI certificates on the server side. Cisco recommends the switch from LEAP to the more secure PEAP [17].

3. EAP-TLS: It establishes a secure connection between the Supplicant and the Authentication Server. It provides mutual authentication by using a X.509 certificate in both sides. Therefore, EAP-TLS requires a PKI to manage the certificates and public keys. EAP-TLS is defined in RFC 2716.

4. EAP-TTLS: It is an authentication method developed by Funk Software and Meetinghouse to replace EAP-TLS. It establishes a secure TLS tunnel between the Supplicant and the Authenticator using a X.509 certificate on the server-side [18]. Once TLS tunnel is established, the client can be securely authenticated using a username/password combination, MS-CHAPv2, or any other authentication mechanism. The advantage of EAP-TTLS over EAP-TLS is that EAP-TTLS doesn’t require the existence of a PKI as only the server will produce a certificate during the authentication process. This eliminates the difficult task of certificate management and simplifies the authentication process on the client-side [17].

5. EAP-MD5: It requires that both the Supplicant and the Authentication Server share a password. The Authentication Server sends a challenge to the Supplicant. The Supplicant creates a hash code of the challenge using MD5 and the shared password and sends it back to the Authentication Server. The Authentication Server
verifies the hash code and render the decision to allow or deny access to the network. EAP-MD5 doesn’t provide mutual authentication and only the Supplicant can be authenticated. Therefore, it can’t detect rouge access points or servers. Moreover, MD5 is a one-way process and can’t be used in key generation. EAP-MD5 is vulnerable to brute force and dictionary attacks and considered insecure [18].
2.2.6 IEEE 802.11R Fast BSS Transition

The IEEE 802.11r standard for Fast BSS Transition, introduced in 2008, is aimed to address the lack of fast and efficient handoff mechanisms in wireless networks [18]. The standard defines a new sublayer within the MAC layer to facilitate fast roaming for the wireless clients using RSN information elements. In addition, it defines a new Authenticated Key Management (AKM) type known as FT-AKM. The standard supports fast and secure roaming within a mobility domain using the Fast BSS Transition (FT) protocol. A mobility domain is a group of Basic Service Sets (BSS) belonging to the same Extended Service Set (ESS).

In a traditional wireless LAN, the client is authenticated by an authentication server using an 802.1X/EAP method. The most common method is EAP-TLS which requires the exchange of nine messages between the client (supplicant) and the authentication server. In some EAP methods, the delay occurs during the authentication process could reach 3-5 seconds. The amount of latency is not tolerable for some real-time applications. This degradation in quality most likely leads to service interruption.

The 802.11r standard adopts the general concept of a decentralized handoff process performed at the network perimeter by the access points rather than the authentication servers. This technique reduces the authentication delay caused by the distance between the client and the authentication server. Moreover, it improves the current key hierarchy described in the 802.1X standard by dividing the Pairwise Master Key (PMK) key, shared between the home access point and the client, into two different levels: PMK-R0 and PMK-R1. The PMK-R0 key is generated by the authentication server using the Master Session Key (MSK) and shared between the client and the designated
key holder (R0KH). The PMK-R1 key is generated by R0KH using the PMK-R0 and shared between the client and the access point. In this case, the access point is appointed as the second key holder (R1KH). The FT protocol uses the following information elements during the handoff exchange:

1. MDIE:

   The Mobility Domain Information Element (MDIE) is distributed by the access point either in the beacon messages or in the probe response message. The MDIE contains the mobility domain identifier and the FT capabilities and policies including the Fast BSS Transition method and the resource reservation protocol.

2. FTIE:

   The Fast Transition Information Element (FTIE) contains important information used during the handoff process including Message Integrity Check (MIC), nonces generated by the client and the access point, and the identifiers of the key holders R0KH and R1KH.

3. RSNIE

   The Robust Security Network Information Element (RSNIE) is similar to the RSNIE used in the 802.11i standard. It contains the identifiers of PMK-R0 and PMK-R1, the cipher type used to secure the session, the Authenticated Key Management (AKM) suit, and other RSN capabilities.
2.2.2.1 FT Key Hierarchy

1. Master Session Key (MSK):

Similar to the IEEE 802.11i security standard, the MSK is the root key in the 802.11r standard. It is mutually generated between the client and the authentication server as a result of the initial authentication using an 802.1X/EAP method. It is used by the client and the authentication to generate the second-level key which is PMK-R0.

2. PMK-R0:

The Pairwise Master Key (PMK) in the 802.11i key hierarchy has been expanded into two different levels in the 802.11r key hierarchy: PMK-R0 and PMK-R1. Each key has a different scope and context defined clearly within the standard. The PMK-R0 key is derived by taking the first 256 bits from the MSK key using a key derivation function as follows:

\[
\text{PMK-R0} = \text{KDF (MSK, "FT-R0", SSID, MDID, R0KH-ID, MAC\text{STA})}
\]

It is shared between the client and the key holder (R0KH). The R0KH can be a designated network with a one-hop secure link to the home access point. Otherwise, the home access point itself can serve as the R0KH for the associated client. The purpose of the PMK-R0 key is to derive the next-level key which is the PMK-R1.

3. PMK-R1:

The PMK-R1 key is generated by the R0KH, which delivers it to the access point associated with the key over a secure channel. The PMK-R1 key is derived using a key derivation function as follows:
PMK-R1 = KDF (PMK-R0, "FT-R1", BSSID, MAC_{STA})

The PMK-R1 is shared between the client and the access point. It is cached in the access point for later use and the access point is designated as the key holder (R1KH). When a handoff is triggered, the client sends a handoff request to the R0KH. In response, the R0KH derives a new PMK-R1 and delivers it to the target access point over a secure channel. The purpose of the PMK-R1 key is to generate the PTK key.

4. Pairwise Transient Key (PTK):

   The PTK, or the session key, is used to encrypt the unicast data frames exchanged between the client and the access point. This key is derived by the client and the associated access point using the PMK-R1 as follows:

   \[ PTK = \text{KDF} (\text{PMK-R1}, \"FT-PTK\", \text{Nonce}_{STA}, \text{Nonce}_{AP}, \text{BSSID}, \text{MAC}_{STA}) \]

Figure 2.5 demonstrates the 802.11r key hierarchy.
2.2.2.2 FT ROAMING METHODS:

The 802.11r standard defines two different methods the client can use to perform a successful handoff with the target access point. These methods are concerned with way the authentication request and response can be exchanged between the client and the target access point. The two methods are: Over-the-Air and Over-the-DS (Distribution System).

1) Over-the-Air:

Using this method, the client communicates the authentication information with the target access directly. The current access point is involved in the FT exchange when the Over-the-Air option is enabled. Unlike the initial authentication using 802.1X/EAP, the materials needed to generate the PMK-R1 is included in the first four-way handshake frames. These frames are the FT authentication request/response
frames and the FT reassociation request and response frames. Figure 2.6 illustrates the handoff process when the Over-the-Air method is used.

The following are the Over-the-Air fast BSS transition steps:

I. The client sends an \textit{Authentication Request} to the target access point. The request contains the FT authentication type, RSNIE, MDIE, and FTIE. The authentication type normally used is the Open Authentication System.

II. The access point sends an \textit{Authentication Response} to the client. The response frame contains the FT authentication algorithm agreed upon with the client, the RSNIE and the MDIE sent previously by the client for confirmation. In addition, the FTIE includes the ID of the R1KH and a nonce generated by the access point.

III. The client responds by sending a \textit{Reassociation Request} to the access point. The request includes the PMKR1-ID and the same MDIE and FTIE sent by the access point in the Authentication Response frame.

IV. The access point sends a \textit{Reassociation Response} frame to the client. The response message contains the same information sent by the client in the Reassociation Request in addition to the Group Transient Key (GTK) shared between the access point and a group of associated clients.

V. Both the client and the access point generate the PTK key as previously described.
2) Over-the-DS

The Over-the-DS is the preferred fast transition method in most implementations. In this method, the client communicates with the target access point through the current access point. When the client decides to roam to a different access point, it sends an FT_Action_Request to the home access point. Next, the home access point forwards the request to the target access point through the distribution system (DS). The authentication request contains information regarding the client, the target access point, the mobility domain, the fast transition protocol, and the security protocol. The target access point
responds by sending an FT_Action_Response message which contains the same information from the FT_Action_Request in addition to the ID of the R1KH key holder.

During the exchange of the first two messages, the client can maintain an active session with the current access point. Once the authentication is done successfully and PMK-R1 is delivered to the target access point, the client can terminate the connection with the home access point. A fast reassociation with the new access point requires the exchange of two messages: Reassociation_Request and Reassociation_Response. The reassociation process here is the exact process performed when using the Over-the-Air method. Figure 2.7 illustrates the handoff process when the Over-the-DS method is used.

The following are the Over-the-DS Fast BSS Transition steps:

I. The client sends an FT_Action_Request to the home access point. The home access point relays the request to the target access point. The request contains STA-ID, TargetAP-ID, RSNIE, MDIE, and FTIE. The RSNIE contains information about the PMK-R0 key shared between the client and the R0KH. The MDIE provides information about the mobility domain as advertised by the target access point. The FTIE contains information used during the fast transition exchange including the R0KH-ID, and a nonce (SNonce) generated by the client to be used during the reassociation phase.

II. The target access point responds by sending an FT_Action_Response frame to the client through the home access point. The response message contains the same information sent previously by the client in the FT_Action_Request in addition to the R0KH-ID. After receiving the
response from the target access point, the R0KH generates a new PMK-R1 key using the PMK-R0 key and delivers it to the target access point.

III. The client dissociates with the home access point and sends a Reassociation_Request to the target access point. The request contains the RSNIE, MDIE, and FTIE. The RSNIE contains information regarding the new PMK-R1 key shared between the client and the target access point. The FTIE contains a newly generated nonce (ANonce) in addition to nonce (SNonce) generated by the client during the authentication phase.

IV. The target access point responds by sending a Reassociation_Response message to the client to confirm the information included in the Reassociation_Request. The response also contains the Group Transient Key (GTK) shared between the access point and the associated clients.

V. The client and the access point mutually generate the Pairwise Transient Key (PTK) using the PMK-R1. The PTK is utilized to establish a secure channel between the two parties.
Figure 2.7 Over-the-DS Fast BSS Transition
2.3 Related Wireless Mesh Networks Research

2.3.1 Performance Analysis of Handoff Delay in WMN

Srivatsa et al. [30] conducted an empirical study to analyze the performance of handoff delay in wireless mesh networks. A simulation based on OPNET developed including a new OPNET model which enables handoff in WMN. The new model has two parts, a Layer 2 model which performs channel scanning and AP association, and a Layer 3 model which is responsible for multi-hop route discovery and location registration.

The study simulated two wireless networks placed in two adjacent buildings. Both networks are connected to each other and the Internet using a mesh router and a mesh gateway. The study tested handoff in both infrastructure mesh networks and client mesh networks. In infrastructure WMN, the study found that the network delay increases from 700ms to 3.5 seconds as the number of hops increases from 1 to 5 hops. This delay is attributed to the signaling message and propagation along the multi-hop path. In client WMN, the handoff delay increases 650ms to around 23 seconds as the number of hops increases from 1 to 5 hops. In this case, the delay is attributed to the long delay in route discovery as the number of hops increases.

Almenares et al [52] is an attempt to measure the overhead of utilizing secure communications in handheld devices. It analyzes the performance of a secure connection between handheld devices and an Apache server by downloading web pages with different file sizes while using different cipher suits such as OpenSSL. It tests symmetric encryption algorithms such as AES, DES, 3DES, and IDEA and asymmetric encryption algorithms such as RSA, DSA, and ECDSA. Although this study provides a detailed analysis of the overhead stemming from using secure communications on handheld
devices, the experiment was conducted using old generation devices which have much lower computational power and battery capacity than the current state-of-art smartphones available for consumers.
2.3.2 Ticket-Based Authentication Protocols

Li et al. [31] propose a ticket-based authentication protocol for wireless mesh networks. This paper suggests two different authentication protocols, one for the initial authentication and the other for the roaming re-authentication. It proposes a trust model based on Kerberos and schemes proposed by Pizada and McDonald [26].

In this scheme, the ticket is used to verify the client ID and can be used to access multiple services. Only one password authentication is needed during the lifetime of the ticket. The tickets are issued and managed by a Ticket Agent (TA) and can be trusted by other entities in the network. The TA establishes trust with other TAs, Mesh Access Points (MAPs), and clients using public key certificates issued by a Certificate Authority (CA). MAP establishes trust with other MAPs using shared symmetric keys. MAPs establish trust with clients using tickets.

There are three types of tickets suggested in this paper:

1. **Client Ticket:** Issued by a TA and used by a client to authenticate itself to the home MAP.

   \[
   T_c = \{ID_c, ID_A, T_{exp}, P_c, Sig_A\}, \text{ where}
   \]

   \(T_{exp}:\) Ticket expiration date and time.

   \(P_c:\) Public key of the client to be used by MAP to verify client’s signature.

   \(Sig_A:\) TA’s signature to verify the authenticity of the ticket.

2. **MAP Ticket:** Issued by a TA and used a MAP to authenticate itself to the client.

   \[
   T_R = \{ID_R, ID_A, T_{exp}, P_R, Sig_A\}.
   \]
3. Transfer Ticket: Issued by the home MAP along with a secret key $K_{MAC}$ and used by the client to authenticate itself to a foreign MAP. The following are the steps to generate a transfer ticket:

a. The client authenticate itself to the home MAP.

b. The home MAP picks a nonce and sends it to the client.

c. Both the client and home MAP generate a shared secret key $K_{MAC}$.

d. The home MAP issues the transfer ticket as follows:

$$\theta_C = \{\mu, V_{K_{MAC}}\}$$

$\mu = \{I_R, I_C, I_A, T_{exp}, MAC_{alg}\}$, where $MAC_{alg}$ is used by a foreign MAP along with $V_{K_{MAC}}$ to verify the authenticity and integrity of the transfer ticket.

This paper proposes two different authentication protocols:

1. Initial login authentication:

1. The MAP periodically broadcasts a beacon message containing its ticket.

2. The client scans for beacon messages to obtain the MAP ticket. It verifies TA’s digital signature using its public key. If verified, the client extracts MAP public key from the ticket and uses it to encrypt message $M_C$ which contains client’s ticket and two nonces $N_{C1}$ and $N_{C2}$.

3. The MAP decrypts message and verifies TA’s digital signature using its public key. If verified, the MAP extracts client’s public key from the ticket and uses it
to encrypt message \( M_C \) which contains client’s ticket and two nonces \( N_{R1} \) and \( N_{R2} \).

4. Both the client and MAP generate the shared MAC key \( K_{MAC} \) by hashing the client’s nonce \( N_{C1} \) and MAP’s nonce \( N_{R1} \).

\[
K_{MAC} = H(N_{C1} || N_{R1})
\]

5. The client uses key \( K_{MAC} \) and MAC algorithm on \( N_{R2} \) to produce a message authentication code \( V_{K_{MAC}}(N_{R2}) \) and sends it to the MAP.

6. MAP computes \( V'_{K_{MAC}}(N_{R2}) \) and if \( V_{K_{MAC}}(N_{R2}) = V'_{K_{MAC}}(N_{R2}) \), the client is authenticated.

7. The MAP repeats steps e and f to authenticate itself to the client.

8. The MAP generates a transfer ticket and sends it to the client.

\[
M = \{V_{K_{MAC}}(N_{C2}), \theta_C\}
\]

2. Handover authentication:

1. The client sends a message to MAP\(_1\) requesting to roam to another MAP.

2. MAP\(_1\) send message \( r = \{ID_c, K_{MAC}\} \) to MAP\(_2\).

3. The client waits for some time and sends the transfer ticket \( \theta_C \) and nonce \( N_C \) to MAP\(_2\).

4. MAP\(_2\) verifies transfer ticket \( \theta_C \) by applying \( K_{MAC} \) from message \( r \) to message \( \mu \) from to generate \( V'_{K_{MAC}}(\mu) \). If \( V'_{K_{MAC}}(\mu) = V_{K_{MAC}}(\mu) \), the transfer ticket \( \theta_C \) is valid.
5. MAP2 verifies client’s ID by using MAC algorithm and KMAC on nonce NC to produce message authentication code \( V'_{K_{MAC}}(N_C) \). If \( V'_{K_{MAC}}(N_C) = V_{K_{MAC}}(N_C) \), the client is authenticated.

6. The client repeats step e to authenticate MAP2.

The handover protocol does not use public keys to make the handover process faster. If MAP2 receive \( \theta_C \) before message r from the client, it cannot verify \( \theta_C \). In this case, MAP2 will send an error message prompting the client to initiate full login authentication. MAP2 propagates message r to its neighbors.

Xu et al. [25] propose a ticket-based handoff authentication protocol for wireless mesh networks. There are two phases for this protocol: ticket generation and re-authentication. During the ticket generation phase, the authentication server (AS) generates multiple tickets for each one-hop neighbor of the current access point (HMR) serving the client (MC) using the Master Key (\( MK_{FMR_i} \)) which is shared between AS and FMRi. Before the tickets can be generated, the HMR must send the IDs of all its one-hop neighbors as well as the client ID to the AS. The AS generates a temporary authentication key TAKi which is shared between the client and FMRi using a keyed hash function

\[
H_{MK_{FMR_i}};
\]

\[
TAKi = H_{MK_{FMR_i}}(ID_{HMR} \mid ID_{FMRi} \mid ID_{MC} \mid N \mid t)
\]

\[
T_i = \{TAKi, ID_{HMR}, ID_{FMRi}, ID_{MC}, N, t\}
\]

The AS delivers all the tickets to the client encrypted using MKMC. Although the tickets are generated using few hash functions, the proactive generation of many tickets that
might not be used at all and the multi-hop delivery to the client might increase the network delay and reduce the throughput.

During the re-authentication phase, the client picks a random number \( r_{MC} \) where \( r_{MC} \in \mathbb{Z}_p^* \) and calculates \( g^{r_{MC}} \) where \( g \) is a generator for \( \mathbb{Z}_p^* \). The client then sends \( \{ g^{r_{MC}}, \text{ID}_{HMR}, \text{ID}_{FMRi}, \text{ID}_{MC}, N, t \} \) to FMR\(_i\). Upon receiving this message, FMR\(_i\) picks a random number \( r_{FMRi} \) which has the same properties of \( r_{MC} \) and calculates \( g^{r_{FMRi}} \). The FMR\(_i\) generates a message \( m_{FMRi} = \{ \text{ID}_{FMRi} \mid g^{r_{MC}} \mid g^{r_{FMRi}} \} \) and sends it to the client along with the keyed hash value of the message using TAK\(_i\). The client verifies the authenticity of the message \( m_{FMRi} \) using TAK\(_i\) and therefore authenticating the FMR\(_i\). The client generates the temporary session key (TSK\(_i\)) = \( (g^{r_{FMRi}})^{r_{MC}} \) which will be used to secure the connection between the client and the FMR\(_i\). The client then sends the message \( m_{MC} = \{ \text{ID}_{MC} \mid g^{r_{MC}} \mid g^{r_{FMRi}} \} \) and its hashed value to the FMR\(_i\). The FMR\(_i\) verifies the message using TAK\(_i\) and subsequently authenticates the client and calculates the TSK\(_i\) = \( (g^{r_{MC}})^{r_{FMRi}} \). Finally, the FMR\(_i\) sends \( \{ \text{ID}_{MC}, t, H(\text{TAK}_i) \} \) to the current home access point (HMR) of the client which will keep it as a record until it expires to verify that it was not used before. After a successful handoff, the client must launch a full authentication with the FMR\(_i\) to finish global handoff and the client, FMR\(_i\), and AS will be sharing a new PMK.

Guansong et al. [32] propose a ticket-based re-authentication protocol for fast handoff in Wireless Local Area Networks (WLANs). In this scheme, once the initial authentication between the client and Authentication Server (AS) is done successfully, the AS generates multiple tickets for each of the neighboring Access Points (APs) to the current AP connected with the client. Each ticket contains the Pairwise Master Key
(PMK_i) generated by AS and is shared between the client and the AP_i. Each PMK_i is encrypted using a Master Key (MK) shared between the AS and the corresponding AP_i.

\[
\text{Ticket} = \{\text{AP}_{MAC}, \text{STA}_{MAC}, (\text{STA}_{MAC} \cdot \text{PMK}_n)_{e_{KAS,AP}}, \text{MIC}(i_k\text{AS,AP})\}
\]

After that, the AS sends all the tickets to the current AP, which in turn unicasts them to the client securely using the Key Encryption Key (KEK).

The client initiates the re-authentication phase by sending a re-authentication request to the Targeted Access Point (TAP) containing the corresponding ticket, a random number (r_S) picked by the client, a random number (r_A) picked by the TAP before the re-authentication phase as a challenge, and a message authentication code MIC derived using PMK_i. The TAP can then verify the ticket using the MIC(i_kAS,AP) and retrieve the PMK_i by decrypting (\text{STA}_{MAC} \cdot \text{PMK}_n)_{e_{KAS,AP}}. TAP then verifies that the client is in possession of PMK_i by applying PMK_i on the MIC of the re-authentication sent by the client. If verified, the TAP has successfully authenticated the client and can now compute the Pairwise Temporal Key (PTK) as:

\[
\text{PTK} = \text{PRF}(\text{AP}_{MAC}, \text{STA}_{MAC}, r_A, r_S, \text{PMK}_i) = (\text{KEK}, \text{KCK}, \text{TK})
\]

Then, TAP sends a re-authentication response to the client containing r_S and MIC_A, which is generated using KCK. The client uses MIC_A to authenticate TAP. This scheme meets some of the important security requirements rendered in this research. First, it provides mutual authentication by allowing both the client and the AP to authenticate each other using the MIC encrypted using a PMK shared between them. Second, using random numbers by both the client and TAP ensures the freshness of the session key (PTK) which is an effective countermeasure against the replay attack and the
man-in-the-middle attack. Third, because this scheme employs random numbers, an attacker cannot derive previously used session keys. This feature is called forward security. Fourth, the use of random numbers prevents attackers from deriving the current session key even with the knowledge of old session key. This feature is called known key security.

Zhang et al. [34] propose two different authentication protocols for wireless mesh networks. First, an initial authentication protocol using a Login-Ticket. It is a two-way handshake scheme rather than the four-way handshake scheme of EAP-TLS. Second, a handoff authentication protocol.

The initial authentication protocol replaces the Pairwise Temporal Key (PTK) generation process, which occurs after the successful authentication of the client by the authentication server (AS). In this scheme, the AS generates a Login-Ticket and sends it to the Access Point (AP) encrypted using the Pairwise Master Key (PMK).

\[
\text{Login-Ticket} = \{ \text{AP}_{\text{MAC}}, \text{STA}_{\text{MAC}}, (\text{STA}_{\text{MAC}} \cdot \text{PMK}_n)_{e_{\text{AS,AP}}}, T_{\text{exp}}, \text{MIC}(i_{\text{AS,AP}}) \}
\]

The AP picks a nonce and sends it to the client along with the encrypted ticket. The client decrypts the ticket using PMK and sends back to the AP along with a nonce picked by the client, the AP’s nonce, \(\text{MIC}_{PMK}\). The AP verifies the ticket using \(T_{\text{exp}}\) and \(\text{MIC}(i_{\text{AS,AP}})\). If the ticket is valid, the AP decrypts \((\text{STA}_{\text{MAC}} \cdot \text{PMK}_n)_{e_{\text{AS,AP}}}\) to retrieve the PMK. Then, it applies the PMK to the Login-Ticket to compute \(\text{MIC}_{PMK^*}\) and compares it to \(\text{MIC}_{PMK}\) to verify that the client is in possession of the PMK. The AP verifies its nonce sent by the client and therefore successfully authenticating the client.

Finally, both the AP and the client generate the PTK as follows:
PTK = PRF (AP\text{MAC}, STA\text{MAC}, A_{nonce}, S_{nonce}, PMK)

= (KEK, KCK, TK)

The AP sends MIC\text{KCK} and S_{nonce} to the client which in term computes MICK\text{KCK}\ast and uses it with S_{nonce} to authenticate the AP. Although this scheme reduces the number of messages exchanged between the client and the AP, it casts the additional burden of decrypting the Login-Ticket on both parties. It requires more computational power and memory space and increases power consumption which is a significant concern especially for battery-operated mobile devices.

The second protocol is a ticket-based fast handover authentication for wireless mesh networks. In this scheme, the AS is not involved in the re-authentication process which reduces the authentication delay significantly. The AP generates a Handover-Ticket for each one of its neighboring APs. The tickets will be used by the client in a one-hop authentication.

Handover-Ticket = \{TAP\text{MAC}, STA\text{MAC}, (STA\text{MAC} \cdot PMKn)_{ek_{CAP\cdot TAP}}, T_{exp},

MIC(i_{k\text{CAP\cdot TAP}})\}

The client initiates the handover process by sending a Re-Association Request message to the Target AP (TAP). The TAP responds Re-Association Response which include ANonce which is used as a challenge. The client sends an EAPOL-Key message containing the Handover-Ticket, SNonce, ANonce, and MIC_{PMKn}. The TAP computes MIC(i_{k\text{CAP\cdot TAP}})\ast and checks the expiry time in order to confirm that the ticket was issued by CAP. It decrypts (STA\text{MAC} \cdot PMKn)_{ek_{CAP\cdot TAP}} to retrieve the PMKn and uses it to compute MIC_{PMKn} \ast which proves that the client has PMKn. Finally, the TAP validates ANonce and therefore authenticating the client. If the client lose association with the
current AP before the handover is completed, a full login authentication must be performed between the client and the TAP.
2.3.3 Public-Key Authentication Protocols

Xiao et al. [33] propose a mutual authentication protocol for roaming in WMN which works on top of 802.1X. It claims to have high security and fewer rounds of interactions compared to other protocols. It achieves that by using different technologies such as hierarchical network topology, three-party key agreement and multi-signature. In this scheme, the network consists of one backbone and one or more local networks. The backbone routers share one database which contains the authorized certificates. There is one off-line Certificate Authority (CA) hosted by an Internet Service Provider (ISP).

This protocol uses Diffie-Hellman (DH) as a three-party key agreement protocol. It takes advantage of the discrete logarithm problem of DH. The following are the steps of DH key agreement between A and B:

1. A large prime number \( p \) and \( g \), which is a generator of the group \( GF(p) \), are picked and published after both parties agree on them.

2. Party A picks a random number \( x \) as the private value and calculates \( Y_a = g^x \mod p \) and sends \( Y_a \) to Party B.

3. Party B picks a random number \( y \) as the private value and calculates \( Y_b = g^y \mod p \) and sends \( Y_b \) to Party A.

4. Party A calculates \( K = (Y_b)^x \mod p \) and Party B calculates \( K' = (Y_a)^y \mod p \)

5. Both parties share the same secret key \( K = K' = g^{xy} \mod p \) which cannot be computed without the values of both \( x \) and \( y \).

The authentication steps between a client a foreign AP (FA) are:
1. The client initiates the authentication process by sending the domain name, p, and $g$ of the Home AP (HA) to the FA.

2. FA generates a Session ID, a random number N, and key material $g^y$ and sends them to the client.

3. The client calculates $g^x$ and $g^{xy}$ and encrypts its ID using HA’s public key. The client appends its signature to the message in order to authenticate itself.

\[
E(e_{HA}.\text{user} \ || \ N)
\]

\[
SIG_c(g^x \ || \ g^{xy} \ || \ E(e_{HA}.\text{user} \ || \ N))
\]

4. FA appends its signature to the message and forwards it to HA.

5. HA verifies FA’s signature and retrieves the message sent by the client. It decrypts the message using its private key and verifies the client’s signature. Then, HA generates key materials $(g^x)^z$ and $(g^y)^z$ and the secret key $K_s = (g^{xy})^z = g^{xyz}$. HA adds its signature to the message and sends it to FA.

\[
M = SID.\ g^{xz}.\ g^{yz}.\ SIG_{HA}(g^{xz} \ || \ g^{xy} \ || \ N))
\]

6. FA verifies HA signature and forwards the message to the client. FA generates the secret key $K_s = (g^{xz})^y = g^{xyz}$.

7. The client verifies HA signature and sends response to FA. The client generates the secret key $K_s = (g^{yz})^x = g^{xyz}$.

8. FA sends the roaming completion response to the client.

This protocol protects client’s privacy by encrypting client ID using the Home AP’s public key. Therefore, only HA has access to client real ID. Moreover, the secret
key materials (x, y, z) and nonce N are generated randomly to improve resilience against some of the well-known attacks such as man-in-the-middle attack and the replay attack. The secret key is generated using a three-party DH key agreement and therefore does not need to be transmitted through the network.

Wei et al. [35] propose an instant authentication scheme for wireless mesh networks using public key cryptography. It provides secure inter-network, intra-network, and inter-domain roaming for the clients. It allows the clients to gain access from the home network or any foreign network.

Each sub-network must have at least one Certificate Authority (CA) acting as the trusted CA for Authentication Centers (AuC), Mesh Access Points (MAP) and clients. Each off-line CA has a public key PK₁ and private key SK₁ pair. Each AuC has its own public key and private key pair in addition to the CA’s public key. Each MAP has a public key and private key pair and the public key of the AuC. Each client has a certificate issued by the off-line CA, a private key, and CA’s public key.

Before a client can roam the network, it must be registered with an AuC in its home network. First, the client sends its certificate to the AuC. If authenticated, the AuC issues a passport to the client.

\[
\text{Passport} = \{\text{Client ID} \parallel \text{Client PK}_c \parallel \text{Client role} \parallel \text{Security Level} \parallel \text{Period of Validity} \parallel \\
\text{Part of inter-network roaming} \parallel \text{Part of intra-network roaming} \parallel \text{AuC}_1 \text{ ID} \parallel \text{AuC}_1 \text{ signature}\}
\]

The AuC sends its public key and the passport to the client as SK₁<<(Passport>> and SK₁<<(PK₁>>. The client verifies the passport using SK₁<<(PK₁>>.
When a client roams between networks within the same domain, it sends its passport to AuC2 of the visited network. The AuC2 verifies the authenticity of the passport using the public key of AuC1, which is SK1<<PK11>>. If verified, the AuC2 will write SK1<<PK12>> into the intra-network part of the passport and add its signature.

As a client moves from a MAP to another within the same network, it sends the passport to new MAP2. MAP2 verifies the passport to authenticate the client using the public key of the AuC1 (PK11). If verified, it sends its public key SK11<<PKMAP2>> to the client. The client can validate the public key using PK12 in SK1<<PK12>>.

When a client moves from another network which resides in a different domain, it sends the certificate issued by CA1 to AuC1 which belongs to the authentication domain of CA2. AuC1 verifies the certificate SK1<<Certificate>> using SK2<<PK1>>. If verified, AuC1 issues a new passport SK21<<Passport>> for the client to be used within the new domain and sends it to the client. Then, the client verifies the passport using SK2<<PK21>> and PK1.

Sultan et al. [24] propose a pairing-based authentication protocol for wireless mesh networks. It provides anonymous roaming by concealing the real ID of the client and supply pseudo IDs. The real ID is known only to the home server. Therefore, only the home server can link past and future activities to a certain client which is called unlinkability.

This scheme operates in four different phases: Home Mesh Gateway setup, roaming agreement, registration of roaming clients, and anonymous authentication.

1. Home MG setup:
When the network is established, the home MG generates its public parameters and secret values. First, it generates $G_1$ and $G_2$, which are two cyclic additive groups of prime order $q$, and $G_T$, which is a cyclic multiplicative group of prime order $q$. Second, the home MG picks two large prime numbers $P$ and $Q$ in addition to some cryptographic hash functions $H_1$, $H_2$, $H_3$, $H_4$, and $H_{4x}$. Third, it picks a master secret key $s \in \mathbb{Z}_q^*$ and a secret value $s' \in \mathbb{Z}_q^*$ where $s \neq s'$. Finally, the home MG computes its public key as: $sQ \in G_2$.

2. Roaming agreement:

First, the foreign MGs send their $ID_{MG_i}^F$ to the home MG. Then, the home MG generates a roaming key for each foreign MG as:

$$K_{roam} = sH_2(ID_{MG_i}^F) \in G_2$$

The home MG sends the roaming keys secret value $s'$ to the corresponding foreign MGs through secure channels.

3. Registration of roaming clients:

After joining the network, every Mesh Node (MN) must register itself with the home MG by sending its ID to the home MG via a secure channel. The home MG generates $K$ unlinkable pseudo identities for the MN.

4. Anonymous authentication:

Each MAP periodically broadcasts beacon messages containing IDMAP, a random number, and the public parameters of the home MG. The steps needed to authenticate the roaming MN are:

a. The MN picks an unused pseudo ID (PID$_{ij}$) and the corresponding secondary key $K_{ij}^{sec}$ and generates a message authentication code MAC as:
\( Aut_1 = MAC_{K_{i,j}^{sec}} (ID_{MG}^H || P_1 || nonce) \)

b. The MN chooses two random numbers \( a, r \in \mathbb{Z}_q^* \) and computes \( aP, aQ_i, V = \left( \frac{1}{a} \right) Q, \) and \( P_1 = rQ_i, \) where \( Q_i = H_1 \left( ID_i \right) \)

c. The MN computes \( h = H_3 \left( P_1 || ID_{MR} || aP || nonce \right) \) and generates the signature \( U = a(r + ah)^{K_{i,prim}}. \) The MN sends nonce, \( Auth_1, aP, P_1, PID_{i,j}, ID_{MG}^H, U, V, aQ_i, \) and \( ID_{MR} \) to the foreign MG.

d. The foreign MG checks the ID of the home MG and uses secret value \( s' \) to compute \( K_{i,j}^{sec} = s'H_1 \left( PID_{i,j} \right) \)

e. The foreign MG computes MAC using \( K_{i,j}^{sec} \) as

\( Aut'_1 = MAC_{K_{i,j}^{sec}} (ID_{MG}^H || P_1 || nonce) \) and checks if \( Aut'_1 = Aut_1 \).

f. The foreign MG verifies signature \( U \) by \( e(U.V) = e \left( P_1 + h'aQ_i, Q_{pub} \right) \),

where

\( h' = H_3 \left( P_1 || ID_{MR} || aP || nonce \right) \)

Tripathy et al. [37] propose a public key authentication and key establishment protocol for wireless mesh networks. This scheme has four different operational phases:

1. System initialization phase:

Before the network is established, an off-line Registration Authority (RA) picks two large prime numbers of 1024-bit size \( (p, q) \) in addition to a small number \( (e) \) as the public key. The secret key is computed as

\[ e \cdot d \equiv 1 \mod (p - 1) (q - 1) \]
The RA makes \( \langle e, H, n, g \rangle \) where \( H \) is a strong one-way hash function, \( n = p \times q \), and \( g \) is a generator of that group. However, RA keeps \( d \) as a secret key.

2. Registration phase:

To join the network, all Access Points (APs) and clients must be registered with the RA. The client sends a registration request to the RA. The RA responds by computing the secret as:

\[
S_c = g^{g_c \times d} \mod n , \quad \text{where} \\
g_c = H (\text{ID}_c \parallel T_{\text{exp}_c})
\]

Then, the RA sends \( \langle S_c, T_{\text{exp}_c} \rangle \) and public parameters \( (e, H, n, g) \) to the client.

3. Authentication and key agreement phase:

a. After the successful execution of the peer link open and confirmation process, the Mesh Router (MR), which acts as the authenticator, generates a nonce \( N_a \) and computes \( \beta_a = g^{N_a} \mod n \).

b. The client generates a nonce \( N_c \) and computes:

\[
\beta_c = g^{N_c} \mod n , \\
\gamma_c = \beta_a^{N_c} \mod n , \text{ and} \\
\alpha_c = S_c^{(N_c + h)} \mod n , \quad \text{where} \quad h = H (\text{ID}_c \parallel \text{ID}_a)
\]

The client sends \( \langle \alpha_c , \beta_c , T_{\text{exp}_c} \rangle \) to the MR.

c. The MR verifies client’s ID and \( T_{\text{exp}} \) by computing

\[
\beta_c^{g \times c} \times g^{h \times g_c} = \alpha_c^e , \quad \text{where} \quad \gamma_a = \beta_a^{N_a} \mod n
\]
If verified, the MR computes $\alpha_a = S_{a}^{(N_a + h)} \mod n$ and sends $<\alpha_a, T_{exp_a}>$ to the client. Also it computes the Pairwise Session Key (PSK) as:

$$K_{AC} = \alpha_{c}^{(e \cdot N_a \cdot g_a)} \cdot \beta_{c}^{(h \cdot g_a \cdot g_c)} \mod n$$

and computes the Pairwise Master Key (PMK) as:

$$K = (ID_a || K_{AC} || ID_c || O)$$

d. The client verifies MR’s ID and $T_{exp}$ by computing

$$\beta_a^{g_a} \cdot g^{h \cdot g_a} = \alpha_a^e$$

If verified, the client computes the PSK as:

$$K_{CA} = \alpha_{a}^{(e \cdot N_c \cdot g_c)} \cdot \beta_{a}^{(h \cdot g_a \cdot g_c)} \mod n$$

and computes the PMK as:

$$K = (ID_a || K_{CA} || ID_c || O)$$

4. Handover authentication phase:

a. When the roaming client moves to another AP, it send a request containing a nonce $N_{A'}$ to the current AP (CAP) for a roaming token. The CAP computes the roaming token as:

$$RT_{AA'} = \{ N_{A'} + 1. \ \eta \}_{K_{AC}}, \text{ Where}$$

$$\eta = \{ N_{A'} + 1. \ ID_c \}_{K_{AA'}}$$

b. The client decrypts the token and validates the token by checking $N_{A'} + 1$. The client sends an association request to the targeted AP (TAP) along with $\eta$ and $H(N_{A'} + 1)$. 

80
c. The TAP decrypts $\eta$ and computes $H(N_{A'} + 1)$. It compares the computed hash value with the one sent by the client to authenticate the client.

This scheme provides some important security features such as mutual authentication, forward secrecy, and resistance to known attacks. The key establishment is considered to be non-trivially secure because a secret key shared between two parties can’t be derived by a third party. Because of the one-way property of the hash function, also called trapdoor, the current session can’t be derived. Also, even if the current session key was compromised, old session keys can’t be obtained without nonces $N_a$ and $N_c$. An attacker is unable to launch a Denial-of-Service (DoS) attack by flooding the AP with a large number of association requests because it will need to provide legitimate credentials.
2.3.4 Identity-Based Authentication Protocols

Wang et al. [36] propose an identity-based authentication protocol for wireless mesh networks called IBS-EAP. It requires the use of a dedicated server as a Private Key Generator (PKG) to generate private keys in addition to the public system parameters $G_1$, $G_2$, $e$, $q$, $p$, $P_{pub}$, $H_1$, and $H_2$ for each node in the network. The dedicated server verifies client’s ID in advance and generates the private key accordingly. The client receives the public parameters and the Authentication Server’s (AS) ID.

In this scheme, the AS has two functions. First, it verifies the client’s ID and validates the PKG system’s public parameters online. Second, it uses its IBS private key to issue access granting tickets to the clients. The ticket contains the client’s ID, a deadline of the service time, and the AS’s signature.

This protocol uses a collection of Diffie-Hellman exchanges, digital signatures, and hash functions for authentication, ID protection, and message integrity. It performs mutual authentication between network nodes based on IBS signature. The public key of a node in the network can be obtained from its ID and the public parameters of the IBS system. Therefore, it eliminates the need to maintain public key certificates and Certificate Revocation Lists (CRL). Instead of utilizing the hierarchical CAs of a Public Key Infrastructure (PKI), it is the task of the PKG to perform IDs verification and issue private keys. It does not require online operations which reduces the cost of system maintenance.

This protocol reduces the computational overhead of both EAP-TLS and EAP-TTLS by requiring less modular exponentiations operations, digital signature verifications, and message authentication code computations. However, this paper failed
to produce any theoretical or empirical results to demonstrate how significant is the
performance improvement over other protocols such as EAP-TLS.
2.3.5 Trust-based Authentication protocols

Xiao et al. [38] propose an authentication protocol for trusted handoff in wireless mesh networks. It is based on the Trusted Platform Module (TPM) hardware and the Trusted Network Connect (TNC) architecture. Any entity wants to gain access to the network must be measured in terms of trust and validated in terms of integrity which include software, hardware, operating systems, and shared libraries.

The TPM module is implemented on a chip integrated into the hardware. The chip is access protected and only the TPM can access the stored data and functions directly using TPM commands. During the boot process, the system generates some hash values to measure the integrity of the platform and verify the settings and configurations. Then, the amassed hash value is stored in TPM in the Platform Configuration Registers (PCR).

The TNC establishes connections based on the configurations stored in the TPM. However, only platforms with verified configurations which meet the security requirements of the network will be considered. Furthermore, the client must verify the security configurations of the Access Point (AP). The TNC defines three types of entities: Access Requester (AR), Policy Enforcement Point (PEP), and Policy Decision Point (PDP). The proposed authentication protocol uses several cryptographic systems such as Elliptic Curve Cryptography (ECC), shared secrets, and three-party key agreement.
CHAPTER III.
HALA: THE HANDOFF AUTHENTICATION LATENCY ASSESSMENT FRAMEWORK

3.1 Overview

The Handoff Authentication Latency Assessment Framework (HALA) is a group of libraries and modules which can be used in the implementation and performance analysis of current and proposed handoff authentication protocol on mobile devices. The HALA framework provides a user-friendly interface to measure the performance of individual or multiple cryptographic methods directly on the devices and receive precise results in an actual implementation. These cryptographic methods, such as hash functions, symmetric encryption, and public key encryption, are the building blocks for any security suite. For example, OpenSSL, an open-source implementation of the Secure Sockets Layer (SSL) and the Transport Layer Security (TLS) protocols, is a security library which is used to establish secure communications between peers. OpenSSL utilizes many cryptographic methods to perform the mutual authentication process between the client and the authentication server, establish the shared keys, and secure the subsequent communications. These methods include digital certificates, public key encryption and decryption, random number generation, and symmetric key encryption and decryption. Therefore, it is important to measure the performance of any security suite on the target devices before developing the actual design to determine if the design elements meet the minimum requirements of the application. It will help us into the evaluation of potential handoff protocols to determine the design trade-offs and compare the performance of these protocols with other benchmarks.
For the purpose of this dissertation, the HALA framework will be utilized in the development of two performance testing tools for the Android and iOS platforms. The tools will be used to analyze the performance of individual cryptographic primitives, as well as some fully developed handoff authentication protocols including our proposed protocols: UFAP and GAP. The framework will be used to evaluate the performance of potential handoff protocols to determine the design trade-offs and compare the performance of these protocols with other benchmarks. The HALA framework is divided into three layers:

1) The Design Layer.

2) The Operation Layer.

3) The Analysis Layer.

Each layer has been designed for specific purposes which include the processing of data collected from certain sources. The output of each layer will be used as an input for the next layer. Figure [15] shows an overview of the layers and the major components of HALA framework.
Figure 3.1 An overview of the HALA framework and its major components

A. The Design Layer

The design layer is responsible for the immediate interaction with the user. It consists of the User Interface, the Protocol Designer, and the Libraries. It provides a user-friendly interface which can be used to build the components of the proposed protocol, view individual results, and analyze aggregated results. In general, the user interface provides some options to control the output received from the other layers.

The protocol designer allows the user to choose several cryptographic methods to be executed in a certain order. For example, the user can choose digital signature verification, a hash function, and a symmetric key encryption to be executed in a sequence and view the final results. The protocol designer can be used to run the cryptographic methods individually. These cryptographic methods are executed on a single mobile device and not in conjunction with other devices.
The libraries are a list of code files and wrapper classes designed to provide easy access to the supported cryptographic methods. In most cases, the process of calling a cryptographic function is very simple. The developer can call a cryptographic function by utilizing an easy-to-use language and passing parameters such as key size and sample data size. The cryptographic methods and parameters used in the development of the HALA framework are listed in table 3.1. The testing tools can be executed on iOS and Android devices.

For iOS devices, the Security Framework and the Common Crypto Library are used for the generation and management of public and private keys, shared secret keys, message authentication codes, and message digests. These cryptographic libraries on iOS devices are developed using APIs available in the Objective C programming language. The Multipeer Connectivity framework can be used to establish and manage mesh links on iOS devices.

For Android devices, the javax.crypto package provides classes and interfaces to implement the same cryptographic applications using Java or C++ programming languages. For the purpose of this dissertation, the Java programming language has been used to develop the HALA framework on Android devices. The open80211s, an open-source implementation of the IEEE 802.11s wireless mesh standard to run on any commodity hardware supported by the Linux kernel, can be used to establish and manage the mesh links on Android devices.
B. The Operation Layer

The Operation Layer is responsible for the execution and data collection of the individual cryptographic methods and security protocols designed in the Design Layer. The Data Collector measures different performance attributes such as CPU usage, memory allocation, execution time, and power consumption for each run. The Data Aggregator compiles the results collected by the Data Collector into a single report and sends it to the Data Analysis Layer.

Table 3.1 List of cryptographic methods and parameters can be utilized in the HALA framework

<table>
<thead>
<tr>
<th>Method</th>
<th>Algorithm</th>
<th>Key Size</th>
<th>Data Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure random numbers</td>
<td>N/A</td>
<td>N/A</td>
<td>8, 16, 32, 64 bytes</td>
</tr>
<tr>
<td>Hash code generation and verification</td>
<td>SHA-256</td>
<td>N/A</td>
<td>50, 100, 200, 500, 1000, 2000, 5000, 10000 KB</td>
</tr>
<tr>
<td>Symmetric encryption and decryption</td>
<td>AES-256</td>
<td>256 bits</td>
<td>128, 192, 256 bytes</td>
</tr>
<tr>
<td>Asymmetric encryption and decryption</td>
<td>RSA-2048</td>
<td>2048 bit</td>
<td>50, 100, 200, 500, 1000, 2000, 5000, 10000 KB</td>
</tr>
<tr>
<td>Digital signature generation and verification</td>
<td>SHA-256,RSA-2048</td>
<td>256 bits, 2048 bits</td>
<td>50, 100, 200, 500, 1000, 2000, 5000, 10000 KB</td>
</tr>
</tbody>
</table>

C. The Analysis Layer

The Analysis Layer is responsible for the verification, compilation, and analysis of the results received from the Operation Layer. The Data Validator receives the compiled report from the Data Aggregator and establishes a baseline by calculating the average execution time of all the runs. Using standard deviation, errant results are excluded from the final report and a message includes the number of errors will be sent to the Operation Layer. The Operation Layer will run the experiment \(n\) times, which is the number of errant results, prepare the new report,
and send it to the Analysis Layer. If the Operation Layer is unable to resolve the issues, an error report will be sent to the Design Layer to inform the user of the problem. The errors can occur due to missing data such as encryption keys, mishandled exceptions, using incorrect parameters, or insufficient memory space.

The Performance Analysis component receives the valid report from the Data Validator and compares the results with the results of running other security protocols and available benchmarks. The results can be compared to the minimum requirements suggested by the user such as maximum latency, memory allocation, and power consumption. However, the current version of the framework only measures the execution time in relation to the maximum latency. The final report which contains the raw data, the comparison analysis, and graphs, if available, will be sent to the Design Layer and displayed to the user.
3.2 Android Security System

Most Android devices available on the market rely on the ARM TrustZone technology, a System-on-Chip (SoC) security architecture, to provide higher levels of protection for the trusted applications and sensitive data. The TrustZone architecture splits a single core of the CPU into two separate virtual cores. Each core represents a so called “world”. The two worlds are: the “normal world” and the “secure world”. The normal world contains the rich OS where non-secure user applications and data exists. The secure world contains a manufacturer-specific Trusted Execution Environment (TEE) where highly sensitive applications and services such as fingerprint verification and online payment can be executed securely.

These two worlds operate independently in separate memory locations. Each process running on the device is distinguished using an extra bit to determine the world in which the process is running. Applications running in the rich OS area do not have direct access to services in the secure area. This distinction adds a layer of security by preventing unauthorized applications and malware from accessing sensitive data and services even in the case of a rooted device.

A. Trusted Execution Environment (TEE)

The Trusted Execution Environment (TEE) protects the integrity, authenticity, and confidentiality of the executed code, data, and runtime states. Unlike, other hardware-based security solutions, TEE can be securely installed and updated [48]. Legitimate user applications running in the rich OS can request services from the TEE using the Client TEE API. For example, a user application can use the Client TEE API to process an online payment or use fingerprints to log into a secure website. There are many available
implementations of the TEE such as Kinibi (Trustonic), T6 (TrustKernel), OP-TEE (Linaro), TLK (Nvidia), and securiTEE (Solacia).

B. Android Encryption Systems

The Android platform includes two different encryption systems: a file-system based encryption system, or full-disk encryption, and a file-based encryption system. The full-disk encryption system is supported Android 5.0 and later versions. However, this feature is not enabled by default and must be activated by the user. On the other hand, the file-based encryption system is supported by Android 7.0 and later versions. It allows users to encrypt single files using different keys.

The full-disk encryption system in Android is designed based on the dm-crypt, a device mapper available in the Linux 2.6 kernel and later [49]. The dm-crypt utilizes the Crypto API to perform the full-disk encryption and decryption processes. The partition containing the user data is encrypted using a single master key. In turn, the master key is protected using the user’s password.

To activate the full-disk encryption system, the user must setup an encryption passcode which will be required to unlock the device at the lock screen. During the secure boot process, a Key Derivation Function (KDF) such as PBKDF2 and SCRYPT is used to derive a symmetric key from the user’s passcode. The derived key is then used to encrypt the actual file encryption master key.

C. Android Secure Storage

Android offers two different classes to store secret credentials such as symmetric keys, asymmetric keys, password, passcodes, and certificates securely on the device. These classes are the KeyChain API and the KeyStore API. Each device has a single keychain
which can be used to store private keys and certificates. The keychain can be accessed from all user applications installed on the device. However, each entry stored in the keychain is password-protected to limit access to authorized applications only.

On the other hand, the KeyStore API offers flexibility and more options for the developers to store secret credentials. It can be used to store different classes of credentials including passwords, passcodes, encryption keys, and certificates. Each entry in the keystore can be protected using a user-provided password. For a user application to retrieve an encryption key from the keystore, it needs to decrypt the file-based keystore and then decrypt the key entry. This two-stages process to retrieve an entry from the keystore could incur a significant delay. This claim will be discussed later in the performance analysis section. Figure 3.2 shows an example of the keystore hierarchy in Android. Finally, Android offers many types of keystores such as BKS, BouncyCastle, PKCS12, and AndroidKeyStore.

Figure 3.2 Keystore hierarchy in Android
3.3 iOS Security System

The iOS security system is a device-specific integration of hardware and software solutions to ensure the confidentiality, integrity, and authenticity of the entire system. It is a trust-based system where all hardware and software components must be validated before the device becomes operable. These components are cryptographically signed by Apple and the signatures can be verified using the Apple Root CA public key embedded in the Boot ROM [50]. During the secure boot process, low-level software such as the bootloader, kernel, and firmware are checked to verify the integrity of the secure boot chain. Any failed step during the secure boot process will prompt the device to operate in the recovery mode.

Imagine an isolated island hiding valuable treasures. An island which is highly protected and is not accessible from the outside by any means. The treasure cannot be located nor collected by anyone from outside the island including those who originally buried it there. The only method to extract bits of that treasure out of the island is by sending a message to the locals via a mailbox located outside the island. The response would be packaged and sent out in a secure manner. In iOS devices, that treasure island is the Secure Enclave.

The Secure Enclave is a coprocessor built into the Apple A-series processors since the release of the A7 processor. It is responsible for the processing and secure storage of sensitive data in iOS devices. It offers some security services to user applications such as online payments and verifying Touch ID fingerprints for easy access. The Secure Enclave Processor (SEP) operates in complete separation from the main processor and uses an encrypted memory space. In addition, SEP has its own operating system which is called
SEPOS, a modified version of the L4 microkernel family. The Secure Enclave communicates with the application processor using a mailbox mechanism.

The Secure Enclave relies on a dedicated hardware AES cryptographic engine to perform all the cryptographic operations requested by the iOS operating system, services, and user applications. The main objective of the AES crypto engine is to offer fast and efficient filesystem encryption and decryption. It is placed between the flash storage and the main memory. The content of the AES crypto engine is not accessible by the operating systems, firmware, services, or user applications. These components can only read the results of the encryption and decryption operations performed by the AES crypto engine.

The secret ingredient used to enhance the iOS protection against unauthorized access is the unique identifier (UID). Each iOS device has its own unique UID which is a per-device AES-256 encryption key. The UID is integrated into the Secure Enclave and the application processor at production. It is important to note that Apple does not keep records of the generated UIDs and therefore it is not possible for the company to gain access to a particular device if it was locked [50].

On the software side, Apple has introduced the Data Protection technology which offers high level security features to system and user applications. The Data Protection API provides four different data protection classes: complete protection, protected unless open, protected until first user authentication, and no protection. Similar protection classes are available to protect keychain entries. Applications must assign a protection class to each file in order to choose the level of protection afforded to each file. If a file is not assigned to a data protection class by the application, the file will be assigned to the protected until first user authentication class by default [50].
The Data Protection API employs a single-file encryption approach to full-disk encryption. When a file is created in iOS, the system generates a per-file AES-256 encryption key associated with that file. Therefore, the system generates and stores an AES-256 encryption key for each file stored on the device. Each data protection class has its own encryption key which is used to protect per-file keys associated with each class. To protect and maintain all encryption keys generated by the system, the Data Protection system incorporates a tree-like key hierarchy and at the top of that hierarchy is the UID.

The main purpose of utilizing the UID in iOS devices is to generate and protect other important encryption keys. In particular, the Secure Enclave derives two AES encryption keys by performing a pre-defined Key Derivation Function (KDF) on a combination of the UID and the user’s passcode. The resulted keys are the file-system master encryption key (EMF) and the Class D key (Dkey).

The EMF key is an encryption key randomly generated during the first installation of iOS. It is stored in a protected space in the flash memory called the Effaceable Storage. The purpose of using the EMF is to encrypt the metadata of all files stored on the device. The metadata of a file contains the actual key used to encrypt that file. This design supports a quick and convenient remote-wipe feature. Rather than wiping the entire file-system to protect the files stored on the device, only erasing the EMF key from the Effaceable storage will redeem all files inaccessible.

When a file is created, it will be encrypted using the associated per-file key. That key then gets wrapped using the corresponding data protection class key and stored in the metadata of the file. The metadata, in turn, is encrypted using the EMF key. The data protection classes’ keys are protected using the UID and user’s passcode. Therefore, the
encrypted files and per-file keys are not affected when the passcode is changed by the user. Only the data protection classes’ keys need to be re-wrapped using the UID and the new passcode. Figure 3.3 shows the iOS key hierarchy.

Figure 3.3 iOS key hierarchy
3.4 Experimental Setup of The HALA Framework on Android and iOS Devices

In this experiment, the focus is on analyzing the performance of the most popular mobile platforms: Android and iOS. According to the International Data Corporation (IDC), Android and iOS claimed Approximately 99% of the global smartphones market share in 2016 [51]. The main objective of this study is to evaluate the performance of recent mobile devices while running cryptographic primitives. The results from this experiment will help researchers and developers to determine the most suitable security protocols for their applications. For example, applications which require short delays or specific deadlines cannot use computationally intensive and time-consuming operations. Therefore, the results from this experiment serve as a baseline for new security protocols and features in upcoming user applications.

The tested cryptographic operations include secure random number generation, hash code generation and verification, symmetric encryption and decryption, asymmetric encryption and decryption, and digital signature generation and verification. The cryptographic algorithms used in this experiment include the Secure Hash Algorithm (SHA-2) as the hash function, the Advanced Encryption Standard (AES-256) as the symmetric encryption algorithm, and RSA-2048 as the public key encryption algorithm.

We developed two performance analysis tools for Android and iOS devices. These tools measure time of execution, power consumption, CPU usage, and memory allocation. For the purpose of this dissertation, the focus will be on the time of execution metric to measure the time needed to perform a single run of a cryptographic operation in milliseconds. The results from this experiment show insignificant impact on the other metrics when performing few cryptographic operations in a short period of time. For
example, a single run of an RSA encryption requires 2MB of memory space and consumes about \%0.01 of the battery. The tested devices were operating in ideal state and only necessary operating system processes were running. No user applications other than the testing tools were active during the experiment.

The performance analysis tools developed for Android and iOS devices perform under the same conditions and use the same parameters. No custom wrapper classes or bridging files were used during the development of these tools. For Android devices, the javax.crypto package provides classes and interfaces to implement different cryptographic primitives using Java. For iOS devices, the Security Framework and the Common Crypto Library were used for the same purpose using Objective-C.

We conducted our performance analysis on six different mobile devices including three iOS devices and three Android devices. The brands and specifications of the tested devices are listed in Table 3.2. We purposefully selected devices from different manufacturers and years of production to examine the improvements made by the smartphone industry with regards to security.
Table 3.2 List of tested mobile devices

<table>
<thead>
<tr>
<th>Device</th>
<th>CPU</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone 6</td>
<td>Apple A8, dual-core 1.4 GHz</td>
<td>1 GB</td>
</tr>
<tr>
<td>iPhone 7</td>
<td>Apple A10, quad-core 2.34 GHz</td>
<td>2 GB</td>
</tr>
<tr>
<td>iPad 4</td>
<td>Apple A6X, dual-core 1.4 GHz</td>
<td>1 GB</td>
</tr>
<tr>
<td>Samsung Galaxy S4</td>
<td>Qualcomm Snapdragon 600, quad-core 1.9 GHz</td>
<td>2 GB</td>
</tr>
<tr>
<td>Samsung Galaxy S7</td>
<td>Qualcomm Snapdragon 820, quad-core 2.2 GHz</td>
<td>4 GB</td>
</tr>
<tr>
<td>OnePlus 3</td>
<td>Qualcomm Snapdragon 820, quad-core 2.2 GHz</td>
<td>6 GB</td>
</tr>
</tbody>
</table>
3.5 Results and Analysis

In this section, a quantitative analysis of the results obtained from the experiments is presented for each of the tested cryptographic methods.

3.5.1 Key Retrieval

In mobile platforms, it is critical to keep sensitive data such as encryption keys and passwords protected from unauthorized access. Keeping such security materials in plain sight exposes applications, data, and potentially the entire device to serious and imminent threats. An application can retrieve encryption keys and passwords from the secure key storage area and store them in its memory space for later use. In some cases, the key retrieval process may incur a significant delay due to the necessary cryptographic functions needed to perform this task. Both Android and iOS utilize different structures to store sensitive data in secure memory locations.

In iOS, each device has a single structure to store sensitive data called the KeyChain. Once the device is unlocked, the keychain is decrypted, and all applications have immediate access to all the entries stored in the KeyChain. However, the entries themselves are individually protected using passwords set by the applications adding another layer of security. In order for an application to access an entry in the keychain, it must decrypt that entry using a password set by the application when the entry was first generated. Our experiment shows that it would take iPad4 about 3 milliseconds and iPhone 7 about 10 milliseconds to retrieve an entry from the keychain.

In Android, applications use the Keystore structure to protect sensitive data. Unlike the single-keychain approach in iOS, each Android application has its own keystore which can only be accessed by that application. Furthermore, each application can create multiple
keystores adding more depth and complexity to the keystore system. To retrieve an entry from a keystore, the application will need to decrypt the selected keystore, search for the specific entry, and then decrypt that entry. Our results show that it would take Samsung Galaxy S7 about 65 milliseconds and Samsung Galaxy S4 about 150 milliseconds to retrieve an entry from the keystore. In this case, the amount of delay caused by fetching the encryption key or password and store it in the user application memory space is not acceptable by some types of applications such as VoIP, videos conferencing, and online gaming. Therefore, it is the conclusion of this research that the use of any cryptographic method which requires the retrieval of sensitive data from the secure memory space is not suitable for the handoff protocols.

Figure 3.4 Execution time of sensitive data retrieval from the secure memory areas on Android and iOS devices in milliseconds
3.5.2 Secure Random Number Generation

The use of randomly generated numbers is an essential element of design in most security protocols. Random Number Generators (RNG) are utilized in most platforms to generate a variety of security materials including symmetric and asymmetric encryption keys, Initialization Vectors (IV), nonces, salts, and padding bytes. Using random numbers adds a layer of unpredictability by reducing the probability of an adversary guessing the output of a function based on prior knowledge of some variables.

In iOS, secure random numbers can be obtained using a cryptographically secure Pseudo Random Number Generator (PRNG) based on the Yarrow algorithm [65]. A 64-bit random value generated by the kernel during the initialization process is used as the seed for the Yarrow algorithm. Applications can read the /dev/random device file to obtain secure random numbers.

In Android, which is built on the Linux kernel, the user applications and the operating system can obtain secure random numbers by reading the /dev/urandom device file. The PRNG collects random numbers from different operating system events such as disk and interruption events and uses the event description as an input in the entropy pool [66]. Applications can retrieve random numbers by invoking the SecureRandom() function available in the java.security package. Figure 3.5 shows the execution times of generating 8-byte and 32-byte secure random numbers on Android and iOS devices in milliseconds. The detailed results of this experiment are listed in Table A.51 and Table A.2 in Appendix A.
3.5.3 Secure Hash Code Generation and Verification

A hash function is any one-way function which transforms arbitrary-sized data into a fixed-sized output. It is a significant cryptographic method which is used in many security applications including message authentication, error detection, digital signature, and cryptographically secure random number generation. A secure hash function is irreversible by design and must be computationally infeasible to compute the original message using a known message digest.

In iOS, the Common Crypto Library supports many hash functions including MD2, MD4, MD5, and the SHA family. However, Apple has ended support for SHA-1 signed certificates used for Transport Layer Security (TLS) in iOS devices. In Android, the java.security package supports a shorter list of hash functions including MD5, and the SHA family. Figure 3.6 shows the execution, measured in milliseconds, of the SHA-256 hash
code generation and verification applied on different file sizes. The detailed results of this experiment are listed in Table A.3 and Table A.4 in Appendix A.

Figure 3.6 Execution time of SHA-2 hash generation and verification on Android and iOS devices in milliseconds
3.5.4 Symmetric Encryption and Decryption

Symmetric encryption or shared-key encryption is a cryptographic method in which a secret value is shared between different parties. The same secret value is used in the encryption and decryption of data. This form of encryption is typically used to securely store and transmit data.

In this experiment, the Advanced Encryption Standard (AES) is used and the key-size is 256 bits. The other key-sizes, 128 bits and 192 bits, have been tested and yielded the same results as the 256 bits key. Therefore, the key-size is not a factor when it comes to the execution time in the AES algorithm. The data sizes used in this experiment are: 50, 100, 200, 500, 1000, 2000, 5000, and 100000 bits. Figure 3.7 shows the execution, measured in milliseconds, of the AES-256 encryption and decryption on different file sizes. The detailed results of this experiment are listed in Table A.5 and Table A.6 in Appendix A.
3.5.5 Asymmetric Encryption and Decryption

The concept of asymmetric encryption was introduced by Whitefield Diffie and Martin Hellman in 1976 [39]. Diffie and Hellman proposed a two-key system, a public key and a private key, where either key can be used for the encryption and decryption of the data. A client can share the public key with others while keeping the private key as a guarded secret. It takes advantage of the unique properties of the arithmetic modular to make it infeasible to deduce the private key by the mere knowledge of the public key. The
RSA algorithm is mostly used in the secure delivery of other sensitive data such as shared-ke

In the experiment, the RSA public key algorithm is used, and the key size is 2048 bits. The data sizes used in this experiment are: 128, 192, and 256 bytes, which the maximum block-size allowed by the RSA algorithm. However, it is worth note that the other available key sizes, 1024 bits and 3072 bits, have been tested and the results are consistent with the 2048 bits key. Therefore, it is the conclusion of this research that the key size does not affect the execution time in the RSA encryption algorithm. Figure 3.8 shows the execution, measured in milliseconds, of the RSA-2048 encryption and decryption. The detailed results of this experiment are listed in Table A.7 and Table A.8 in Appendix A.
Figure 3.8 Execution time of RSA-2048 encryption and decryption on Android and iOS devices in milliseconds

3.5.6 Digital Signature Generation and Verification

Digital signature is a common cryptographic method used to preserve the authenticity and integrity of the original message in transit. A message can be signed using the private key and verified on the other side of the communication using the corresponding public key. In this case the sender cannot deny sending the message because only an entity with access to the private key can issue the digital signature.

In this experiment, the SHA-2 algorithm is used as the hash function and the RSA-2048 algorithm is used as the public key algorithm. To generate the digital signature, first,
the intended file is hashed using the SHA-2 algorithm. Second, the resulted hash code is signed using the private key. To verify the digital signature, first, the signature is decrypted using the corresponding public key and the result will be the original hash code. Second, the file is hashed by the same hash function used by the sender. Third, the resulted hash code is compared with the hash code retrieved from the digital signature. If the hash codes match, the received file is authenticated. Figure 3.9 shows the execution, measured in milliseconds, of the digital signature generation and verification using SHA-2 and RSA-2048. The detailed results of this experiment are listed in Table A.9 and Table A.10 in Appendix A.

Figure 3.9 Execution time of digital signature generation and verification on Android and iOS devices in millisecond
CHAPTER IV.
UFAP: ULTRA-FAST HANDOFF AUTHENTICATION PROTOCOL

4.1 Introduction

In this section, we propose a new token-based handoff authentication protocol for wireless mesh networks. This particular process occurs when a client, which has been fully authenticated by an authentication server, roams between access points. When the client initiates a connection with another access point, it must be re-authenticated to the network through the new access point.

In our protocol, we reduce the authentication delay by delegating the authentication authority to the edge access points rather than the authentication server. This design eliminates the network delay caused by the multi-hop authentication needed when using a centralized authentication scheme. Utilizing a single-hop authentication reduces the number of authentication messages travelling throughout the network. Only first-time full authentication requests go to the authentication server while roaming requests can be directly processed by the access points. In addition, our protocol needs only three authentication messages exchanged between the client and the foreign access point (FAP) to perform mutual authentication.
4.2 Design Principles for Handoff Protocols in WMN

Before we discuss our proposed protocol, we outline the main design principles that should be incorporated in any secure authentication scheme:

A. Mutual authentication: Both the client and the access point must be able to mutually verify their authenticity. It is an effective countermeasure against well-known attacks such as rogue access points and man-in-the-middle attack.

B. Traffic management: The number of authentication messages exchanged during the handoff authentication should be minimized. Additionally, the number of multi-hop transmissions must be limited. This feature effectively preserves the network capacity and minimizes the delay [15].

C. Seamless roaming: In WMN, the clients are assumed to be highly mobile and can be associated with different access points while being connected to the same network. A fast and efficient handoff protocol should be implemented to reduce the handoff delay and to avoid service interruption. Delegating the authentication process to the access points rather than the centralized authentication server is a viable solution. It eliminates the propagation between the client and the authentication server.

D. Privacy and Anonymity: The real identities of the clients, the access points, the routers, and the authentication servers must be protected. This condition can be satisfied by: 1) encrypt the headers of all the messages which contain the real identities. 2) use temporary pseudo identities which can be changed after first use or after a predetermined period of time.
E. Resilience against known attacks: The open nature of the shared wireless medium and the lack of physical security leaves WMN vulnerable to many serious attacks. The most common attacks against wireless networks are passive eavesdropping, message injection, man-in-the-middle, masquerading, and rogue access points. The authentication protocol must prevent these attacks by implementing countermeasures such as mutual authentication, data confidentiality, message integrity and key freshness.

F. Power efficiency: Mobile devices rely on batteries, a limited source of power, to maintain their operational status. Therefore, authentication protocols designed for mobile devices should power wisely. The number of cryptographic operations performed during the authentication process should be minimized and intensive computations should be performed by the access points or the authentication servers.

4.3 Handoff Protocol Design

Our Ultra-Fast Authentication Protocol (UFAP) protocol is a fast, secure, and efficient handoff authentication protocol for WMN using pre-distributed parameters and a token. UFAP is not a substitute to the initial full authentication process which occurs when the client first joins the network. The client must be authenticated by the authentication server to be able to use UFAP for faster roaming. Here, both the client and the authentication server must provide valid credentials to perform mutual authentication using an authentication protocol such as EAP-TLS. Once the client is fully authenticated, UFAP can
be utilized in a one-hop handoff authentication exchange. Figure 2 shows the message exchanges in EAP-TLS and UFAP.

![Figure 4.1 Handoff authentication in WMN using UFAP and EAP-TLS](image)

In WMN, a fully authenticated client and the Home Access Point (HAP) establish a secure link using a shared Pairwise Master Key (PMK) generated by the authentication server during the initial authentication process. Therefore, all data exchanged between them after this point are assumed to be protected using a set of shared keys. Once the secure link is established, the HAP creates a record \( M \) of handoff parameters for the client which will be used later during the handoff authentication. Each handoff record includes the following parameters:

1. A pair of pseudo client IDs: \( ID_{C1} \) and \( ID_{C2} \), to obscure the real identity of that client.

\( ID_{C1} \) is the client roaming ID used by the client to communicate with Foreign Access
Points (FAP) during the handoff exchange. On the other hand, ID\textsubscript{C2} is used by the authenticated clients to remain anonymous while sending regular traffic to the network. However, ID\textsubscript{C2} is not included in the authentication token because it must be changed constantly which will require frequent updates to the handoff records (\(M\)).

2. A different nonce (Nonce\textsubscript{H}) for each client.

3. Other parameters include ID\textsubscript{HAP}, ID\textsubscript{AS}, and timestamp \(t\).

The HAP sends the resulted handoff record (\(M\)) to the client and all one-hop neighboring access points over already established secure links where:

\[
M = \{\text{ID\textsubscript{HAP}}, \text{Nonce\textsubscript{H}}, \text{ID\textsubscript{C1}}, \text{ID\textsubscript{AS}}, t\}
\]

When the handoff is triggered, the client can initiate a mutual authentication sequence with the FAP as follows:

1. The client generates a handoff request (\(M_1\)) which contains a nonce (Nonce\textsubscript{C}), generated by the client, ID\textsubscript{HAP}, ID\textsubscript{C1}, and a timestamp \(t\).

\[
M_1 = \{\text{ID\textsubscript{HAP}}, \text{ID\textsubscript{C1}}, \text{Nonce\textsubscript{C}}\}
\]

2. Now, the FAP searches its database for record \(M\) pertaining to the client using ID\textsubscript{HAP} and ID\textsubscript{C1} and retrieves Nonce\textsubscript{H} from that record.

3. The FAP XOR Nonce\textsubscript{H} and Nonce\textsubscript{C}, and generates a hash of the result. Next, it sends the response (\(M_2\)) which contains the hash and a nonce (Nonce\textsubscript{F}) generated by the FAP to the client.

\[
M_2 = \{\text{Nonce\textsubscript{F}}, \text{H(Nonce\textsubscript{C} } \oplus \text{ Nonce\textsubscript{H}})\}
\]
where H is a secure hash function.

4. The client generates a hash of the XOR result of $Nonce_H$ and $Nonce_C$ and compares it to the hash sent by the FAP in authenticating the FAP. Only access points with access to the valid handoff record have knowledge of $Nonce_H$ and therefore can generate the correct hash.

5. The client XOR $Nonce_H$ and $Nonce_F$ and generates a hash of the result and sends the hash ($M_3$) to the FAP.

\[ M_3 = \{H(Nonce_F \oplus Nonce_H)\} \]

6. The FAP generates a hash of the XOR result of $Nonce_H$ and $Nonce_F$ and compares it to the hash sent by the client concluding a successful mutual handoff authentication attempt. The FAP is now the new HAP of the client.

7. The client can promptly resume sending and receiving regular network traffic through the new HAP. The message will be secured using the session key shared between the client and the old HAP. The new HAP can act as a message relay until a new session key is mutually computed between the client and the new HAP.

8. The client and the new HAP can mutually generate a new session key using a key derivation function such as the Password-Based Key Derivation Function 2 (PBKDF2). In this case, $ID_{HAP}$, $Nonce_H$, $Nonce_C$, and $Nonce_F$ can be used as the passphrase as follows:

\[
\text{Session-Key} = (PRF, \text{Passphrase}, \text{salt}, \text{iterations}, \text{key-length})
\]

Where Passphrase = \{ID_{HAP} \| Nonce_H \| Nonce_C \| Nonce_F\),
and PRF is a pseudorandom function such HMAC−SHA1

Figure 4.2 demonstrates the handoff record generation phase and the handoff authentication phase.

Figure 4.2 Token generation and handoff authentication using UFAP
4.4 Implementation and Experimental Setup

We conducted an experiment on a WMN to draw a comparison between our proposed protocol UFAP and other currently deployed protocols such as EAP-TLS and EAP-PEAP. We analyzed the performance of these protocols based on execution time and network efficiency. The experiment was performed in an actual setup using a small-scale network shown in Figure 4.3.

The network consists of five wireless routers, an authentication server, and three clients. The network is designed to simulate a multi-hop wireless connection between an authentication server and a client. We configured the routers to operate as access points within a Wireless Distribution System (WDS). In our experiment, each router is directly connected to one or two other routers. For example, Router2 is directly connected to Router1 and Router3 but not Router4. In addition, all the links between the nodes within the network are fully wireless.

We used five Linksys WRT54GL Wireless-G routers to form the backbone network. These routers have the same hardware version. Each router has a Broadcom BCM4710 processor with a 125 MHz MIPS32 core, 16 MB RAM, and 4 MB of flash memory. We installed the DD-WRT router firmware, an open source platform for routers, on each router and used the WDS feature to establish and manage network connectivity.

The authentication server is a Dell PC with an Intel i7-3612 CPU, 8GB RAM, and 750GB of storage. The server runs the latest version of the Ubuntu Server operating system. We installed FreeRADIUS, an open source RADIUS server, to perform user authentication using EAP-TLS and EAP-PEAP. In addition, the authentication server has been used to
generate and distribute self-signed Certificate Authority (CA), server, and client
certificates used during the authentication process.

On the client side, we test the authentication protocols, subject of this experiment,
on different types of mobile devices. The selected devices, which include a smartphone, a
tablet, and laptop, covers a wide spectrum of performance, mobility, and application
profiles. The smartphone used here is a Samsung Galaxy S7 which has a Qualcomm
Snapdragon 820 chip with a 2200 MHz quad-core processor, 4GB RAM, and 3000mAh in
battery capacity. The tablet is an iPad 4 which has an Apple A6X chip with a 1400MHz
dual-core processor, 1GB RAM, and 11,560mAh in battery capacity. The laptop is a
MacBook Air which features a 1.8 GHz Intel Core i5 processor, 8GB RAM, and runs
macOS Sierra. The specifications of the devices used in this experiment are listed in Table
4.1

Figure 4.3 Experimental setup to test the performance of UFAP, EAP-TLS, and EAP-PEAP
in a multi-hop WMN

119
Table 4.1 A detailed list of the devices used in this experiment

<table>
<thead>
<tr>
<th>Device</th>
<th>Type</th>
<th>CPU</th>
<th>Memory</th>
<th>OS</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linksys WRT54GL</td>
<td>Router (1)</td>
<td>Broadcom BCM4710</td>
<td>16 MB</td>
<td>DD-WRT</td>
<td>192.168.1.1</td>
</tr>
<tr>
<td>Linksys WRT54GL</td>
<td>Router (2)</td>
<td>Broadcom BCM4710</td>
<td>16 MB</td>
<td>DD-WRT</td>
<td>192.168.1.2</td>
</tr>
<tr>
<td>Linksys WRT54GL</td>
<td>Router (3)</td>
<td>Broadcom BCM4710</td>
<td>16 MB</td>
<td>DD-WRT</td>
<td>192.168.1.3</td>
</tr>
<tr>
<td>Linksys WRT54GL</td>
<td>Router (4)</td>
<td>Broadcom BCM4710</td>
<td>16 MB</td>
<td>DD-WRT</td>
<td>192.168.1.4</td>
</tr>
<tr>
<td>Linksys WRT54GL</td>
<td>Router (5)</td>
<td>Broadcom BCM4710</td>
<td>16 MB</td>
<td>DD-WRT</td>
<td>192.168.1.5</td>
</tr>
<tr>
<td>Dell PC</td>
<td>Authentication server</td>
<td>Intel i7-3612</td>
<td>8 GB</td>
<td>Ubuntu Server + FreeRADIUS</td>
<td>192.168.1.10</td>
</tr>
<tr>
<td>Samsung Galaxy S7</td>
<td>Client (1)</td>
<td>Snapdragon 820</td>
<td>4 GB</td>
<td>Android</td>
<td>192.168.1.101</td>
</tr>
<tr>
<td>iPad 4</td>
<td>Client (2)</td>
<td>Apple A6X</td>
<td>1 GB</td>
<td>iOS</td>
<td>192.168.1.102</td>
</tr>
<tr>
<td>MacBook Air</td>
<td>Client (3)</td>
<td>Intel Core i5 1.8 GHz</td>
<td>8 GB</td>
<td>macOS Sierra</td>
<td>192.168.1.103</td>
</tr>
</tbody>
</table>
4.5 Performance Analysis

In this section, we compare the performance of our protocol against EAP-TLS and EAP-PEAP. We measure the performance of these protocols using the following metrics: 1) the number of authentication messages, 2) the distance between the client and the authentication authority, 3) the number of the cryptographic operations needed.

First, our protocol reduces the number of authentication messages exchanged between the client and the access point to only three messages. This is a major enhancement over EAP-TLS which requires the transfer of nine messages during the entire authentication exchange [11]. The number of authentication messages travelling through the network affects network throughput. The network can become congested by the high volume of authentication messages when a large number of clients try to authenticate, or re-authenticate, themselves to the network. Figure 4.4 illustrates the volume of authentication message with the increased number of clients.

![Figure 4.4 Number of authentication messages in UFAP, EAP-TLS, and EAP-PEAP](image.png)
Second, current authentication protocols, such as EAP-TLS and EAP-PEAP, require the existence of a centralized authentication system managed by a dedicated authentication server. In this case, the authentication delay would increase with each additional hop between the client and the authentication server. Our protocol utilizes a decentralized design for the authentication process by delegating the authentication authority to the access points rather than the dedicated authentication server. We use a vouch system where the current home access point can attest to the legitimacy of the client using predetermined values. This feature effectively eliminates the authentication delay stemming from the multi-hop transfer of authentication messages.

Third, our scheme significantly reduces the overhead of using computationally intensive cryptographic methods in authentication. In particular, we use a secure cryptographic hash algorithm such as SHA-1 or SHA-256 which is faster and less resource exhaustive alternative to symmetric and asymmetric encryption algorithms. Our protocol requires the execution of two hash generation and two hash verification functions during the entire authentication session. On the other hand, EAP-TLS and EAP-PEAP require the use of computationally expensive methods such digital certificates, symmetric and asymmetric encryption.

The results from our experiment confirm that our protocol (UFAP) is faster than the current authentication protocols. In average, it takes between 3-14 ms to authenticate the client while using UFAP. On our implementation, even in real-time applications, which require a maximum delay between 50-150 ms, service will be resumed without experiencing any degradation or interruption. This is a significant improvement over the performance of EAP-TLS and EAP-PEAP which take an average of 2 seconds and 800
ms, respectively, to perform the same task. Therefore, our protocol provides seamless roaming and an enhanced user experience without depleting the limited resources of the client. The new protocol prevails in comparison with existing protocols such as EAP-TLS and EAP-PEAP. Our experiment shows that UFAP is 500 times faster than EAP-TLS and 250 times faster than EAP-PEAP while generating fewer authentication messages. Figure 4.5 demonstrates a performance comparison between UFAP, EAP-TLS, and EAP-PEAP using three different mobile devices.

![Figure 4.5 Runtime comparison between UFAP, EAP-TLS, and EAP-PEAP on different mobile devices](image-url)
4.6 Security Analysis

In this section, we discuss the security features embedded in the design of our protocol (UFAP). We use a scenario-based analysis to examine the resilience of UFAP against common threats and attacks. We considered a wide variety of well-known attacks such as man-in-the-middle, replay attack, message injection, and rogue access points.

1. Attacks Against Secure Hash Functions

In a recent development, Google Research and CWI Amsterdam announced that they have successfully devised the first practical collision attack on SHA-1. It is a variant of the identical-prefix collision attack in which a fixed prefix can be extended by two block pairs to create a collision regardless of the suffix such that:

\[ H(P||M_1^{(1)}||M_2^{(1)}||S) = H(P||M_1^{(2)}||M_2^{(2)}||S) \]

This attack requires computational effort of approximately \(2^{63.1}\) SHA-1 compressions to find two collided messages which have the same hash. The computational power needed to find a collision using this attack is estimated to be 100 GPU years [17]. To put this into perspective, this attack, albeit practical, would require more than 1.5 trillion GPU cores working simultaneously for less than 3 milliseconds to break our scheme, not factoring in data transmission time. We implemented a countermeasure in our protocol specifically to prevent this type of attack by avoiding the use of known prefixes, such as nonces exchanged publicly. Instead, our protocol hashes the XOR result of these nonces as follows:

\[ H(\text{NonceC} \oplus \text{NonceH}) \text{ and } H(\text{NonceF} \oplus \text{NonceH}) \]
Therefore, our protocol is computationally secure even when SHA-1, which has been deprecated by NIST since 2011, is the hashing algorithm of choice. However, we recommend using more secure hashing algorithms such as SHA-2, which we tested in our experiment, and SHA-3. Both algorithms have no currently known weaknesses [16].

2. Message Interception

Message $M_1$, sent from the client to the FAP for the purpose of initiating the handoff authentication phase, contains $Nonce_C$ which is used along with $Nonce_H$ to authenticate the FAP. Although $Nonce_C$ is sent in the clear, an adversary eavesdropping will not be able to generate the correct challenge by merely intercepting $M_1$. The challenge hash can only be produced with prior knowledge of both $Nonce_C$ and $Nonce_H$. This scheme relies on the secrecy of $Nonce_H$, a nonce generated by the HAP, which is pre-distributed over secure links to the client and all one-hop FAPs. Therefore, the authentication process is still secure even though some important data are sent in plain-text. The same analysis can be drawn in the case of $M_2$ interception, which contains $Nonce_F$ generated by the FAP to authenticate the client.

3. Forward and Backward Secrecy

The UFAP protocol provides strong backward and forward secrecy due to the consistent and continuous use of fresh elements. In forward secrecy, challenge hashes produced during past authentication attempts are always protected, in case current challenges have been compromised. Each time handoff authentication is triggered, a fresh set of unrelated secure random nonces are used. Therefore, an adversary in possession of current session keys will not be able to deduce previous session keys. In backward secrecy,
current session keys are protected with the use of fresh secure random numbers even if a past session key has been obtained by an adversary.

4. Privacy

To protect the real identity of the client, a one-time pseudo identity (ID$_{C1}$) generated by the HAP can only be used during the handoff authentication. Once the client is authenticated, ID$_{C1}$ must be discarded and another pseudo identity (ID$_{C2}$) is used during the regular session. ID$_{C2}$ can be changed by the HAP after a predetermined period of time or after the exchange of a certain number of messages.
CHAPTER V.
GAP: THE GLOBAL HANDOFF AUTHENTICATION PROTOCOL

5.1 GAP Protocol Design

The Global Authentication Protocol (GAP) is a ticket-based handoff authentication protocol intended to be utilized in wireless mesh networks (WMN) and similar wireless networks. Using GAP, the handoff is performed through a one-hop exchange between the client and the target access point (TAP). Therefore, it does not require the involvement of a third-party such as an authentication server or even the home access point (HAP). Instead, it uses available information provided by the ticket issuer to execute the handoff sequence successfully. The needed information during the handoff include the client roaming ID (IDC), the ticket issuer ID (IDT), and some global parameters.

The global parameters used in the GAP protocol are G1 and G2. These parameters are generated, maintained, and distributed to all the authenticated nodes within the network by the authentication server. The global parameters are used among other information to deduce the encryption key used to encrypt the roaming ticket by the ticket issuer. Hence, any legitimate node with authorized access to the global parameters can instantly authenticate the roaming client.

Figure 5.1 shows the content of the handoff ticket as follows:

1) The ticket ID, which can used to confirm the authenticity of the ticket by contacting the ticket issuer.
2) The client roaming ID (IDC), which is a unique value generated by the ticket issuer and can only be used by the client during the handoff exchange. A different IDC is used with each new roaming ticket. IDC is a component of the later discussed roaming token.

3) Ticket issuer ID (IDT).

4) A nonce (NonceT) generated by the ticket issuer. Each roaming ticket has a different NonceT. This nonce is used during the handoff process for mutual authentication between the client and the target access point (TAP).

5) A timestamp (t) which is used as a countermeasure against known attacks such as man-in-the-middle attack and the replay attack. In addition, the timestamp is used to determine the life-span of the ticket. An expired roaming ticket will be replaced by a new one produced by the ticket issuer with a new IDC.

<table>
<thead>
<tr>
<th>Ticket ID 6 bytes</th>
<th>Client Roaming ID 6 bytes</th>
<th>Ticket Issuer ID 6 bytes</th>
<th>NonceT 32 bytes</th>
<th>Timestamp 12 bytes</th>
</tr>
</thead>
</table>

Figure 5.1 GAP Handoff ticket format

The roaming ticket is encrypted using an encryption key (K) generated by the ticket issuer. The encryption key is the result of an XOR combination of the client roaming ID, the ticket issuer ID, the global parameter G1, and the roaming token as follows:

\[ K = \{IDC \oplus IDT \oplus G1 \oplus \text{Roaming\_Token}\} \]

The roaming token is the resulted hash code of the client roaming ID, the ticket issuer ID, and the global parameter G2 as follows:

\[ \text{Roaming\_Token} = \{IDC \oplus IDT \oplus G2\} \]
5.2 Handoff Process

When the handoff is triggered, the following handoff sequence, demonstrated in Figure 5.2, is initiated by the client:

1- The client generates a handoff request (M1). The handoff request contains the roaming ticket, the client roaming ID (IDC), the home access point ID (IDHAP), a nonce (NonceC) generated by the client, and a timestamp (t) as follows:

   \[ M1 = \{ IDC, IDHAP, NonceC, t, roaming ticket \} \]

   The client then sends M1 to the target access point (TAP).

2- The TAP extracts the encrypted roaming ticket from M1 and deduces the encryption key (K) using information from M1 and the global parameters as follows:

   a. The TAP generates the roaming token by hashing the XOR combination of IDC, IDT, and G2.

   b. The TAP generates the encryption key K’ and decrypt the roaming ticket.

   \[ K' = \{ IDC \oplus IDT \oplus G1 \oplus \text{roaming token} \} \]

3- Upon the successful decryption of the roaming ticket, the TAP retrieves NonceT from the ticket. This particular nonce is known only to the ticket issuer and the client. The TAP performs an XOR operation on NonceT and NonceC, received from the client in M1, and hashes the result using a secure hash function. The resulted hash code is sent to the client along with a nonce (NonceF) generated by the TAP as follows:

   \[ M2 = \{ NonceF, H(NonceT \oplus NonceC) \} \]

129
4- The client generates a hash of the XOR result of $Nonce_H$ and $Nonce_C$ and compares it to the hash sent by the FAP in authenticating the FAP. Only access points with access to the valid handoff record have knowledge of $Nonce_H$ and therefore can generate the correct hash.

5- The client XOR $Nonce_H$ and $Nonce_F$ and generates a hash of the result and sends the hash ($M_3$) to the FAP.

$$M_3 = \{H(Nonce_F \oplus Nonce_H)\}$$

6- The FAP generates a hash of the XOR result of $Nonce_H$ and $Nonce_F$ and compares it to the hash sent by the client concluding a successful mutual handoff authentication attempt. The FAP is now the new HAP of the client.

7- The client can promptly resume sending and receiving regular network traffic through the new HAP. The message will be secured using the session key shared between the client and the old HAP. The new HAP can act as a message relay until a new session key is mutually computed between the client and the new HAP.

8- The client and the new HAP can mutually generate a new session key using a key derivation function such as the Password-Based Key Derivation Function 2 (PBKDF2). In this case, $ID_{HAP}$, $Nonce_H$, $Nonce_C$, and $Nonce_F$ can be used as the passphrase as follows:

$$\text{Session-Key} = (\text{PRF}, \text{Passphrase}, \text{salt}, \text{iterations}, \text{key-length})$$

Where Passphrase = \{ID_{HAP} || Nonce_H || Nonce_C || Nonce_F\},

and PRF is a pseudorandom function such as HMAC–SHA1.
Figure 5.2 Ticket generation and the subsequent handoff process using GAP
5.3 Performance Analysis

The Global Authentication Protocol (GAP) is a ticket-based handoff authentication protocol for WMN. It is a variation of the Ultra-Fast Authentication Protocol (UFAP), discussed in the previous section, designed for the hybrid type of WMN. In this scheme, the wireless client can be connected to the network through a dedicated access point or through any authenticated client with authorized access to the network. The median client serves as an access point and authenticator for the new client.

In this section, we compare the properties and performance of the GAP protocol against UFAP, EAP-TLS, and EAP-PEAP. The comparison is drawn based on the following:

1) Keys and handoff records management:

In the UFAP protocol, the home access point generates a single handoff record for each client and delivers to all one-hop neighboring access points. This procedure requires that the access point maintain adequate allocated memory space to store the handoff records. Moreover, UFAP is ideal for networks with good stability and limited topology changes. The clients are assumed to have linear movement patterns and most likely to roam between neighboring access points. A sudden change in the network topology caused by failed access points or routers, or unusual movement pattern could lead the client to seek reassociation with an access point which does not possess the handoff record. In this case, the access point will not be able to recognize the client and will initiate a full authentication with the authentication server. GAP will more useful in this situation than UFAP.
Using the GAP protocol, the home access generates a single handoff ticket for each client. This protocol does not require the pre-distribution of handoff records to the neighboring access points. Instead, the handoff ticket can be recognized and accessed by any access point or clients authorized to serve as access points. The encrypted handoff ticket can be decrypted with a key generated on-the-fly using information available to all authenticated members. The needed information includes the global parameters generated and distributed by the authentication server, the client roaming identifier generated by the home access point, and the ticket issuer identifier.

2) Security overhead:

Maintaining a certain security level in wireless network may require high computational overhead. Unlike EAP-TLS and other 802.1X/EAP authentication methods, GAP does not require the use of digital certificates to perform a mutual authentication between the client and the access point. In EAP-TLS, for example, the client and the authentication server need to exchange their respective digital certificates in order to establish a secure channel between them. After that, a symmetric key is generated by the authentication server and delivered to the client through using the asymmetric keys. The newly delivered key will be used to generate other keys that will be used to secure the communications between the client and the authentication server on one hand, and between the client and the access point on the other. In total, four asymmetric and six symmetric functions are executed during the EAP-TLS authentication not counting the secure links between the intermediate routers.
This method of authentication requires intensive computational power and exhibits an appetite for limited resources such as CPU, memory, and battery. The same criticism can be cast on the EAP-PEAP as it requires the establishment of a TLS tunnel during to secure the authentication session. As a result, the authentication delay caused by the use of an 802.1X/EAP authentication method during the handoff will not be tolerated by most application types and will most likely cause service interruption. For example, an authentication using EAP-TLS, with 3-5 hops between the client and the authentication server, will take between 1-3 seconds depending on how capable the client is to handle the computational requirements of this method. Old smartphones with low CPU power and low memory space can take more than 5 seconds to finish the authentication process. This amount of delay is not acceptable by VoIP applications, for example, which require a maximum delay of 150 ms. Figure 5.4 shows the runtime comparison between UFAP, GAP, EAP-TLS, and EAP-PEAP.

The GAP protocol promises the same level of security advertised in the 802.11i security standard with far less security overhead. It requires the execution of only one symmetric decryption process, which is performed only by the access points and not the clients. In addition, a total of four hash generation and verification functions need to be executed during the handoff, one by the client and one by the access point. This scheme significantly reduces the delay occurs over the course of the entire handoff exchange. The results from the experiment, described in the previous chapter, shows that delay when using GAP protocol in infrastructure WMN network is between 6 - 20 ms. This number can increase in
the case of joining a hybrid WMN if the authenticator is another client with limited capabilities. In this case, the estimated delay is between 12 - 26 ms. Figure 5.3 shows the runtime comparison between UFAP, GAP when used in the Infrastructure WMN, and GAP when used in the Hybrid WMN.

Figure 5.3 Runtime comparison between UFAP and GAP

Figure 5.4 Runtime comparison between UFAP, GAP, EAP-TLS, and EAP-PEAP on different mobile devices
CHAPTER VI.
CONCLUSION

6.1 Conclusion

In this dissertation, we have designed and implemented two light-weight, fast, and efficient handoff authentication protocol for WMN. The new protocols provide fast roaming for the clients by performing a one-hop handoff process at the access points level rather than a multi-hop and computationally expensive exchange with the centralized authentication servers. An experiment was conducted on recent mobile devices to determine which cryptographic primitives can be used as the main components in the design of the new protocols. We concluded that the use of symmetric and asymmetric encryption on the client-side is not suitable for handoff protocols due the heavy computations and the long execution times incurred. In our protocols, we use the less expensive secure hash functions and symmetric encryption on the access point side.

We have demonstrated that our protocols prevail in comparison with existing protocols such as EAP-TLS and EAP-PEAP and provide better performance, QoS, and user experience especially for real-time applications such as VoIP. Our handoff latency assessment framework, HALA, was used during the experiment to measure the performance our protocols in comparison with EAP-TLS and EAP-PEAP. Our experiment, conducted on a small-scale testbed, shows that our protocols, UFAP and GAP, are 500 times faster than EAP-TLS and 250 times faster than EAP-PEAP while generating fewer authentication messages. We have proved that both UFAP and GAP provide adequate security countermeasures against well-known attacks such as the man-in-the-middle and the replay attack by using securely generated random numbers, pseudo...
identifiers, and timestamps. These countermeasures are implemented to insure freshness in the handoff materials and to prevent the malicious reuse of the expired handoff records and handoff tickets. Attacks aimed to exploit the weaknesses of secure hash function, an important in securing both protocols, have been discussed and countered using simple yet effective methods such as the XOR operation. Our dual-ID system continuously protects the privacy of the client by using periodically changed pseudo-identifiers while concealing the real ones. In Particular, the use of a pseudo identifier generated specifically for the handoff exchange helps in protecting the client and the authentication process as an adversary will not be able to recognize that the client has acquired new identity.
6.2 Future Work

In the future, we plan to expand the scope of the proposed handoff authentication protocols to explore the possibility of designing an inter-network handoff protocol. One potential approach is to study the possibility of integrating the GAP protocol with MobileIP (MIP), a standard developed by the Internet Engineering Task Force (IETF), in which the client is able to roam between different networks while maintaining the same home IP address. For example, information provided by the home agent and the foreign agent can be included in the main structure of the handoff ticket.

Other future research directions include the study of secure routing protocols in WMN. This is a significant issue especially for a technology which features high mobility, dynamic topology, and high scalability. Most routing protocols utilized in WMN such as DSR, AODV, and OLSR operate under the assumption of a trusted environment in which the routing process is protected against malicious nodes. Protection against known routing protocols attacks such as routing table poisoning, wormhole attack, Denial of Service (DoS), and black hole attack was not incorporated in the design of these protocols. Instead, these protocols rely on the security mechanisms and services such as Intrusion Detection (ID) and authentication to provide the needed protection.
REFERENCES


APPENDIX A

EXPERIMENTAL RESULTS

Table A.1 Execution time of 8-bytes secure random number generation on Android and iOS devices in nanoseconds

<table>
<thead>
<tr>
<th>Runs</th>
<th>Galaxy S4</th>
<th>Galaxy S7</th>
<th>OnePlus 3</th>
<th>iPhone 7</th>
<th>iPad 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,024,583</td>
<td>110,053</td>
<td>316,563</td>
<td>168,625</td>
<td>462,958</td>
</tr>
<tr>
<td>2</td>
<td>1,444,291</td>
<td>143,750</td>
<td>273,073</td>
<td>181,000</td>
<td>496,933</td>
</tr>
<tr>
<td>3</td>
<td>1,202,958</td>
<td>139,895</td>
<td>210,937</td>
<td>183,458</td>
<td>503,682</td>
</tr>
<tr>
<td>4</td>
<td>1,086,708</td>
<td>119,884</td>
<td>171,667</td>
<td>179,791</td>
<td>493,614</td>
</tr>
<tr>
<td>5</td>
<td>1,411,250</td>
<td>138,021</td>
<td>286,979</td>
<td>205,916</td>
<td>565,340</td>
</tr>
<tr>
<td>6</td>
<td>966,750</td>
<td>121,406</td>
<td>299,971</td>
<td>190,375</td>
<td>522,672</td>
</tr>
<tr>
<td>7</td>
<td>1,234,437</td>
<td>126,902</td>
<td>294,818</td>
<td>174,813</td>
<td>479,946</td>
</tr>
<tr>
<td>8</td>
<td>1,323,625</td>
<td>141,823</td>
<td>242,005</td>
<td>182,229</td>
<td>500,308</td>
</tr>
<tr>
<td>9</td>
<td>1,144,833</td>
<td>129,890</td>
<td>191,302</td>
<td>181,625</td>
<td>498,648</td>
</tr>
<tr>
<td>10</td>
<td>1,248,979</td>
<td>128,953</td>
<td>229,323</td>
<td>192,854</td>
<td>529,477</td>
</tr>
</tbody>
</table>
Table A.2 Execution time of 16-bytes secure random number generation on Android and iOS devices in nanoseconds

<table>
<thead>
<tr>
<th>Runs</th>
<th>Galaxy S4</th>
<th>Galaxy S7</th>
<th>OnePlus 3</th>
<th>iPhone 7</th>
<th>iPad 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,135,917</td>
<td>165,625</td>
<td>301,145</td>
<td>201,250</td>
<td>604,812</td>
</tr>
<tr>
<td>2</td>
<td>1,017,708</td>
<td>135,157</td>
<td>199,011</td>
<td>194,458</td>
<td>584,400</td>
</tr>
<tr>
<td>3</td>
<td>1,142,001</td>
<td>132,135</td>
<td>147,552</td>
<td>185,250</td>
<td>556,728</td>
</tr>
<tr>
<td>4</td>
<td>1,021,719</td>
<td>145,521</td>
<td>214,947</td>
<td>195,125</td>
<td>586,405</td>
</tr>
<tr>
<td>5</td>
<td>1,007,500</td>
<td>119,115</td>
<td>211,719</td>
<td>198,916</td>
<td>597,798</td>
</tr>
<tr>
<td>6</td>
<td>1,158,417</td>
<td>117,135</td>
<td>349,844</td>
<td>175,625</td>
<td>527,802</td>
</tr>
<tr>
<td>7</td>
<td>1,128,644</td>
<td>125,839</td>
<td>231,929</td>
<td>192,758</td>
<td>579,291</td>
</tr>
<tr>
<td>8</td>
<td>1,207,385</td>
<td>134,772</td>
<td>210,488</td>
<td>195,719</td>
<td>588,190</td>
</tr>
<tr>
<td>9</td>
<td>1,223,361</td>
<td>138,194</td>
<td>220,184</td>
<td>189,501</td>
<td>569,503</td>
</tr>
<tr>
<td>10</td>
<td>1,192,476</td>
<td>129,362</td>
<td>234,923</td>
<td>199,442</td>
<td>599,378</td>
</tr>
</tbody>
</table>
Table A.3 Execution time of 32-bytes secure random number generation on Android and iOS devices in nanoseconds

<table>
<thead>
<tr>
<th>Runs</th>
<th>Galaxy S4</th>
<th>Galaxy S7</th>
<th>OnePlus 3</th>
<th>iPhone 7</th>
<th>iPad 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,246,434</td>
<td>214,633</td>
<td>299,219</td>
<td>212,458</td>
<td>667,417</td>
</tr>
<tr>
<td>2</td>
<td>1,966,466</td>
<td>187,884</td>
<td>261,928</td>
<td>241,083</td>
<td>757,340</td>
</tr>
<tr>
<td>3</td>
<td>2,845,087</td>
<td>271,831</td>
<td>378,958</td>
<td>255,666</td>
<td>803,151</td>
</tr>
<tr>
<td>4</td>
<td>2,380,168</td>
<td>227,410</td>
<td>317,032</td>
<td>229,291</td>
<td>720,296</td>
</tr>
<tr>
<td>5</td>
<td>1,527,343</td>
<td>145,928</td>
<td>203,438</td>
<td>224,000</td>
<td>703,675</td>
</tr>
<tr>
<td>6</td>
<td>2,000,093</td>
<td>191,097</td>
<td>266,407</td>
<td>231,125</td>
<td>726,058</td>
</tr>
<tr>
<td>7</td>
<td>2,106,450</td>
<td>201,258</td>
<td>280,574</td>
<td>226,771</td>
<td>712,378</td>
</tr>
<tr>
<td>8</td>
<td>2,405,777</td>
<td>229,857</td>
<td>320,443</td>
<td>248,375</td>
<td>780,245</td>
</tr>
<tr>
<td>9</td>
<td>2,612,628</td>
<td>249,621</td>
<td>347,995</td>
<td>242,479</td>
<td>761,724</td>
</tr>
<tr>
<td>10</td>
<td>1,953,755</td>
<td>186,669</td>
<td>260,235</td>
<td>226,646</td>
<td>711,986</td>
</tr>
</tbody>
</table>
Table A.4 Execution time of 64-bytes secure random number generation on Android and iOS devices in nanoseconds

<table>
<thead>
<tr>
<th>Runs</th>
<th>Galaxy S4</th>
<th>Galaxy S7</th>
<th>OnePlus 3</th>
<th>iPhone 7</th>
<th>iPad 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,691,455</td>
<td>283,674</td>
<td>364,624</td>
<td>239,208</td>
<td>804,992</td>
</tr>
<tr>
<td>2</td>
<td>2,627,403</td>
<td>276,923</td>
<td>355,947</td>
<td>245,000</td>
<td>824,483</td>
</tr>
<tr>
<td>3</td>
<td>2,863,754</td>
<td>301,834</td>
<td>387,966</td>
<td>270,791</td>
<td>911,276</td>
</tr>
<tr>
<td>4</td>
<td>2,941,849</td>
<td>310,065</td>
<td>398,546</td>
<td>244,541</td>
<td>822,939</td>
</tr>
<tr>
<td>5</td>
<td>2,503,444</td>
<td>263,858</td>
<td>339,153</td>
<td>228,166</td>
<td>767,833</td>
</tr>
<tr>
<td>6</td>
<td>2,726,162</td>
<td>287,332</td>
<td>369,326</td>
<td>240,875</td>
<td>810,602</td>
</tr>
<tr>
<td>7</td>
<td>2,588,607</td>
<td>272,834</td>
<td>350,691</td>
<td>252,388</td>
<td>849,346</td>
</tr>
<tr>
<td>8</td>
<td>3,045,646</td>
<td>321,005</td>
<td>412,608</td>
<td>268,142</td>
<td>902,362</td>
</tr>
<tr>
<td>9</td>
<td>2,927,209</td>
<td>308,522</td>
<td>396,563</td>
<td>265,982</td>
<td>895,093</td>
</tr>
<tr>
<td>10</td>
<td>2,787,168</td>
<td>293,762</td>
<td>377,591</td>
<td>273,673</td>
<td>920,975</td>
</tr>
</tbody>
</table>
Table A.5 Execution time of SHA-2 hash code generation on Android and iOS devices in nanoseconds

<table>
<thead>
<tr>
<th>Size (KB)</th>
<th>Galaxy S4</th>
<th>Galaxy S7</th>
<th>OnePlus 3</th>
<th>iPhone 7</th>
<th>iPad 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1,022,574</td>
<td>310,167</td>
<td>348,021</td>
<td>311,220</td>
<td>423,216</td>
</tr>
<tr>
<td></td>
<td>1,228,566</td>
<td>261,385</td>
<td>343,645</td>
<td>305,331</td>
<td>419,212</td>
</tr>
<tr>
<td></td>
<td>1,074,072</td>
<td>282,511</td>
<td>306,875</td>
<td>309,748</td>
<td>422,215</td>
</tr>
<tr>
<td></td>
<td>1,151,319</td>
<td>271,948</td>
<td>325,260</td>
<td>307,539</td>
<td>420,714</td>
</tr>
<tr>
<td></td>
<td>1,138,444</td>
<td>263,401</td>
<td>295,495</td>
<td>307,907</td>
<td>420,964</td>
</tr>
<tr>
<td></td>
<td>1,125,570</td>
<td>254,854</td>
<td>265,729</td>
<td>308,275</td>
<td>421,214</td>
</tr>
<tr>
<td>100</td>
<td>4,409,375</td>
<td>987,864</td>
<td>909,115</td>
<td>972,500</td>
<td>1,332,041</td>
</tr>
<tr>
<td></td>
<td>4,158,000</td>
<td>1,049,740</td>
<td>1,380,104</td>
<td>979,541</td>
<td>1,338,402</td>
</tr>
<tr>
<td></td>
<td>4,137,021</td>
<td>880,677</td>
<td>1,001,824</td>
<td>975,646</td>
<td>1,334,862</td>
</tr>
<tr>
<td></td>
<td>4,147,511</td>
<td>965,209</td>
<td>1,190,964</td>
<td>977,593</td>
<td>1,336,632</td>
</tr>
<tr>
<td></td>
<td>4,006,089</td>
<td>869,349</td>
<td>1,001,824</td>
<td>975,646</td>
<td>1,337,157</td>
</tr>
<tr>
<td></td>
<td>3,864,667</td>
<td>773,490</td>
<td>1,094,532</td>
<td>978,791</td>
<td>1,337,683</td>
</tr>
<tr>
<td>200</td>
<td>6,936,167</td>
<td>1,422,708</td>
<td>1,773,907</td>
<td>1,563,625</td>
<td>2,474,000</td>
</tr>
<tr>
<td></td>
<td>6,776,583</td>
<td>1,556,042</td>
<td>1,339,791</td>
<td>1,550,708</td>
<td>2,471,223</td>
</tr>
<tr>
<td></td>
<td>6,557,230</td>
<td>1,164,368</td>
<td>1,605,496</td>
<td>1,563,896</td>
<td>2,476,876</td>
</tr>
<tr>
<td></td>
<td>6,666,906</td>
<td>1,360,205</td>
<td>1,472,643</td>
<td>1,557,302</td>
<td>2,474,050</td>
</tr>
<tr>
<td></td>
<td>6,422,599</td>
<td>1,133,116</td>
<td>1,454,864</td>
<td>1,560,734</td>
<td>2,476,901</td>
</tr>
<tr>
<td></td>
<td>6,178,292</td>
<td>906,027</td>
<td>1,094,532</td>
<td>978,791</td>
<td>1,337,683</td>
</tr>
<tr>
<td>500</td>
<td>15,539,334</td>
<td>1,776,094</td>
<td>2,619,323</td>
<td>3,611,458</td>
<td>5,880,625</td>
</tr>
<tr>
<td></td>
<td>15,895,292</td>
<td>2,123,490</td>
<td>2,261,354</td>
<td>3,403,125</td>
<td>5,884,391</td>
</tr>
<tr>
<td></td>
<td>15,677,730</td>
<td>1,831,094</td>
<td>2,525,938</td>
<td>3,493,208</td>
<td>5,885,599</td>
</tr>
<tr>
<td></td>
<td>15,786,511</td>
<td>1,977,292</td>
<td>2,393,646</td>
<td>3,448,167</td>
<td>5,884,995</td>
</tr>
<tr>
<td></td>
<td>15,801,318</td>
<td>1,931,693</td>
<td>2,413,099</td>
<td>3,411,562</td>
<td>5,887,783</td>
</tr>
<tr>
<td></td>
<td>15,816,125</td>
<td>1,886,094</td>
<td>2,432,552</td>
<td>3,374,958</td>
<td>5,890,572</td>
</tr>
<tr>
<td>1000</td>
<td>30,236,583</td>
<td>2,674,531</td>
<td>3,593,229</td>
<td>6,871,750</td>
<td>12,323,916</td>
</tr>
<tr>
<td></td>
<td>29,881,833</td>
<td>2,858,333</td>
<td>3,204,167</td>
<td>6,681,750</td>
<td>12,329,146</td>
</tr>
<tr>
<td></td>
<td>30,034,625</td>
<td>2,576,797</td>
<td>4,213,151</td>
<td>6,731,208</td>
<td>12,324,902</td>
</tr>
<tr>
<td></td>
<td>29,958,229</td>
<td>2,717,565</td>
<td>3,708,659</td>
<td>6,706,479</td>
<td>12,327,024</td>
</tr>
<tr>
<td></td>
<td>29,895,447</td>
<td>2,598,314</td>
<td>4,270,866</td>
<td>6,648,573</td>
<td>12,326,455</td>
</tr>
<tr>
<td></td>
<td>29,832,666</td>
<td>2,479,063</td>
<td>4,833,073</td>
<td>6,590,666</td>
<td>12,325,887</td>
</tr>
<tr>
<td>2000</td>
<td>33,684,500</td>
<td>5,940,677</td>
<td>6,243,385</td>
<td>10,671,500</td>
<td>25,728,166</td>
</tr>
<tr>
<td></td>
<td>33,691,624</td>
<td>5,301,146</td>
<td>5,495,573</td>
<td>11,250,708</td>
<td>25,723,247</td>
</tr>
<tr>
<td></td>
<td>34,915,750</td>
<td>6,333,620</td>
<td>6,406,927</td>
<td>11,096,188</td>
<td>25,725,345</td>
</tr>
<tr>
<td></td>
<td>34,303,687</td>
<td>5,817,383</td>
<td>5,951,250</td>
<td>11,173,448</td>
<td>25,724,296</td>
</tr>
<tr>
<td></td>
<td>35,225,344</td>
<td>6,271,973</td>
<td>6,260,860</td>
<td>11,347,161</td>
<td>25,723,409</td>
</tr>
<tr>
<td></td>
<td>36,147,000</td>
<td>6,726,563</td>
<td>6,570,469</td>
<td>11,520,875</td>
<td>25,722,523</td>
</tr>
<tr>
<td>Size (KB)</td>
<td>Galaxy S4</td>
<td>Galaxy S7</td>
<td>OnePlus 3</td>
<td>iPhone 7</td>
<td>iPad 4</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>5000</td>
<td>93,978,125</td>
<td>6,278,215</td>
<td>13,398,948</td>
<td>14,549,616</td>
<td>32,484,758</td>
</tr>
<tr>
<td></td>
<td>82,752,792</td>
<td>7,218,177</td>
<td>12,779,323</td>
<td>15,291,041</td>
<td>33,016,475</td>
</tr>
<tr>
<td></td>
<td>88,214,459</td>
<td>6,502,422</td>
<td>13,860,646</td>
<td>15,002,308</td>
<td>32,498,711</td>
</tr>
<tr>
<td></td>
<td>85,483,625</td>
<td>6,860,300</td>
<td>13,319,985</td>
<td>15,146,675</td>
<td>32,757,593</td>
</tr>
<tr>
<td></td>
<td>83,967,209</td>
<td>6,793,431</td>
<td>13,821,164</td>
<td>15,300,837</td>
<td>32,635,128</td>
</tr>
<tr>
<td></td>
<td>82,450,792</td>
<td>6,726,563</td>
<td>14,322,344</td>
<td>15,455,000</td>
<td>32,512,664</td>
</tr>
<tr>
<td>10000</td>
<td>166,063,708</td>
<td>24,099,740</td>
<td>23,969,427</td>
<td>22,442,208</td>
<td>47,211,645</td>
</tr>
<tr>
<td></td>
<td>161,866,458</td>
<td>13,966,771</td>
<td>28,110,782</td>
<td>24,726,416</td>
<td>47,221,872</td>
</tr>
<tr>
<td></td>
<td>164,432,833</td>
<td>21,722,709</td>
<td>24,795,703</td>
<td>22,907,667</td>
<td>47,218,639</td>
</tr>
<tr>
<td></td>
<td>163,149,646</td>
<td>17,844,740</td>
<td>26,453,243</td>
<td>23,817,041</td>
<td>47,220,256</td>
</tr>
<tr>
<td></td>
<td>162,975,802</td>
<td>18,595,208</td>
<td>26,037,611</td>
<td>23,595,083</td>
<td>47,222,945</td>
</tr>
<tr>
<td></td>
<td>162,801,958</td>
<td>19,345,677</td>
<td>25,621,979</td>
<td>23,373,125</td>
<td>47,225,634</td>
</tr>
</tbody>
</table>
Table A.6 Execution time of SHA-2 hash code verification on Android and iOS devices in nanoseconds

<table>
<thead>
<tr>
<th>Size (KB)</th>
<th>Galaxy S4</th>
<th>Galaxy S7</th>
<th>OnePlus 3</th>
<th>iPhone 7</th>
<th>iPad 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>3,641,750</td>
<td>467,500</td>
<td>916,250</td>
<td>1,840,916</td>
<td>2,298,625</td>
</tr>
<tr>
<td></td>
<td>3,811,583</td>
<td>822,969</td>
<td>1,115,521</td>
<td>1,792,541</td>
<td>2,238,222</td>
</tr>
<tr>
<td></td>
<td>3,710,500</td>
<td>699,635</td>
<td>957,526</td>
<td>1,821,229</td>
<td>2,269,729</td>
</tr>
<tr>
<td></td>
<td>3,761,042</td>
<td>761,302</td>
<td>1,036,524</td>
<td>1,806,885</td>
<td>2,253,975</td>
</tr>
<tr>
<td></td>
<td>3,770,146</td>
<td>846,536</td>
<td>1,017,663</td>
<td>1,804,213</td>
<td>2,247,404</td>
</tr>
<tr>
<td></td>
<td>3,779,250</td>
<td>931,770</td>
<td>998,802</td>
<td>1,801,541</td>
<td>2,240,832</td>
</tr>
<tr>
<td>200</td>
<td>6,870,250</td>
<td>651,979</td>
<td>1,076,458</td>
<td>2,397,416</td>
<td>2,936,458</td>
</tr>
<tr>
<td></td>
<td>6,529,208</td>
<td>1,073,906</td>
<td>1,152,448</td>
<td>2,389,458</td>
<td>2,946,238</td>
</tr>
<tr>
<td></td>
<td>6,772,542</td>
<td>973,620</td>
<td>1,066,120</td>
<td>2,394,229</td>
<td>2,936,435</td>
</tr>
<tr>
<td></td>
<td>6,650,875</td>
<td>1,023,763</td>
<td>1,109,284</td>
<td>2,391,843</td>
<td>2,938,874</td>
</tr>
<tr>
<td></td>
<td>6,662,855</td>
<td>1,159,511</td>
<td>1,082,532</td>
<td>2,391,442</td>
<td>2,936,411</td>
</tr>
<tr>
<td></td>
<td>6,674,834</td>
<td>1,295,260</td>
<td>1,055,781</td>
<td>2,390,041</td>
<td>2,936,411</td>
</tr>
<tr>
<td>500</td>
<td>15,842,625</td>
<td>1,411,832</td>
<td>2,589,896</td>
<td>4,487,041</td>
<td>6,397,625</td>
</tr>
<tr>
<td></td>
<td>15,252,542</td>
<td>2,122,163</td>
<td>1,863,958</td>
<td>4,248,500</td>
<td>6,756,833</td>
</tr>
<tr>
<td></td>
<td>15,681,042</td>
<td>1,667,349</td>
<td>2,012,553</td>
<td>4,422,979</td>
<td>6,464,901</td>
</tr>
<tr>
<td></td>
<td>15,466,792</td>
<td>1,894,756</td>
<td>1,938,255</td>
<td>4,335,739</td>
<td>6,610,867</td>
</tr>
<tr>
<td></td>
<td>15,493,126</td>
<td>1,908,810</td>
<td>1,686,732</td>
<td>4,347,328</td>
<td>6,571,521</td>
</tr>
<tr>
<td></td>
<td>15,519,459</td>
<td>1,922,865</td>
<td>1,435,209</td>
<td>4,358,916</td>
<td>6,532,176</td>
</tr>
<tr>
<td>1000</td>
<td>29,584,542</td>
<td>2,435,104</td>
<td>2,747,709</td>
<td>7,404,875</td>
<td>12,941,833</td>
</tr>
<tr>
<td></td>
<td>30,056,000</td>
<td>3,056,458</td>
<td>2,399,479</td>
<td>7,522,375</td>
<td>12,739,681</td>
</tr>
<tr>
<td></td>
<td>29,701,542</td>
<td>2,448,724</td>
<td>2,329,922</td>
<td>7,469,833</td>
<td>12,876,289</td>
</tr>
<tr>
<td></td>
<td>29,878,771</td>
<td>2,752,591</td>
<td>2,364,701</td>
<td>7,496,104</td>
<td>12,807,985</td>
</tr>
<tr>
<td></td>
<td>29,848,656</td>
<td>2,607,468</td>
<td>2,138,418</td>
<td>7,515,448</td>
<td>12,809,364</td>
</tr>
<tr>
<td></td>
<td>29,818,541</td>
<td>2,462,344</td>
<td>1,912,135</td>
<td>7,534,791</td>
<td>12,810,744</td>
</tr>
<tr>
<td>2000</td>
<td>35,192,834</td>
<td>5,457,136</td>
<td>5,105,365</td>
<td>12,662,000</td>
<td>26,168,125</td>
</tr>
<tr>
<td></td>
<td>33,415,667</td>
<td>5,842,294</td>
<td>3,365,105</td>
<td>12,772,250</td>
<td>25,942,242</td>
</tr>
<tr>
<td></td>
<td>47,520,209</td>
<td>4,905,235</td>
<td>4,029,844</td>
<td>12,057,646</td>
<td>26,081,746</td>
</tr>
<tr>
<td></td>
<td>40,467,938</td>
<td>5,373,764</td>
<td>3,697,475</td>
<td>12,414,948</td>
<td>26,011,994</td>
</tr>
<tr>
<td></td>
<td>50,157,760</td>
<td>4,863,549</td>
<td>3,325,899</td>
<td>11,934,119</td>
<td>26,003,680</td>
</tr>
<tr>
<td></td>
<td>59,847,583</td>
<td>4,353,333</td>
<td>2,954,323</td>
<td>11,453,291</td>
<td>25,995,367</td>
</tr>
<tr>
<td>5000</td>
<td>82,309,834</td>
<td>8,598,490</td>
<td>12,164,896</td>
<td>22,715,291</td>
<td>66,187,166</td>
</tr>
<tr>
<td></td>
<td>81,953,125</td>
<td>6,461,094</td>
<td>6,738,490</td>
<td>19,044,541</td>
<td>67,365,112</td>
</tr>
<tr>
<td></td>
<td>82,720,209</td>
<td>10,091,120</td>
<td>12,233,516</td>
<td>21,816,916</td>
<td>66,282,804</td>
</tr>
<tr>
<td></td>
<td>82,336,667</td>
<td>8,276,107</td>
<td>9,486,003</td>
<td>20,430,729</td>
<td>66,823,958</td>
</tr>
<tr>
<td></td>
<td>82,733,626</td>
<td>9,929,929</td>
<td>10,894,069</td>
<td>20,674,635</td>
<td>66,601,200</td>
</tr>
<tr>
<td></td>
<td>83,130,584</td>
<td>11,583,750</td>
<td>12,302,135</td>
<td>20,918,541</td>
<td>66,378,442</td>
</tr>
<tr>
<td>Size (KB)</td>
<td>Galaxy S4</td>
<td>Galaxy S7</td>
<td>OnePlus 3</td>
<td>iPhone 7</td>
<td>iPad 4</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>10000</td>
<td>161,362,875</td>
<td>19,103,473</td>
<td>24,569,896</td>
<td>25,549,709</td>
<td>124,657,166</td>
</tr>
<tr>
<td></td>
<td>165,270,000</td>
<td>17,313,229</td>
<td>23,882,396</td>
<td>25,350,708</td>
<td>123,889,231</td>
</tr>
<tr>
<td></td>
<td>165,453,542</td>
<td>19,104,185</td>
<td>24,255,001</td>
<td>24,594,521</td>
<td>124,466,994</td>
</tr>
<tr>
<td></td>
<td>165,361,771</td>
<td>18,208,707</td>
<td>24,068,698</td>
<td>24,972,615</td>
<td>124,178,112</td>
</tr>
<tr>
<td></td>
<td>167,452,989</td>
<td>18,656,801</td>
<td>24,004,402</td>
<td>24,305,974</td>
<td>124,227,467</td>
</tr>
<tr>
<td></td>
<td>169,544,208</td>
<td>19,104,896</td>
<td>23,940,105</td>
<td>23,639,333</td>
<td>124,276,821</td>
</tr>
</tbody>
</table>
Table A.7 Execution time of AES-256 encryption on Android and iOS devices in nanoseconds

<table>
<thead>
<tr>
<th>Size (KB)</th>
<th>Galaxy S4</th>
<th>Galaxy S7</th>
<th>OnePlus 3</th>
<th>iPhone 7</th>
<th>iPad 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>16,145,542</td>
<td>4,067,709</td>
<td>2,071,979</td>
<td>21,012,541</td>
<td>44,418,027</td>
</tr>
<tr>
<td></td>
<td>17,056,958</td>
<td>3,964,823</td>
<td>3,972,239</td>
<td>24,161,750</td>
<td>44,398,273</td>
</tr>
<tr>
<td></td>
<td>16,561,105</td>
<td>4,074,636</td>
<td>3,058,490</td>
<td>21,447,708</td>
<td>44,441,615</td>
</tr>
<tr>
<td></td>
<td>16,809,031</td>
<td>4,019,730</td>
<td>3,515,364</td>
<td>22,804,729</td>
<td>44,419,944</td>
</tr>
<tr>
<td></td>
<td>16,892,849</td>
<td>4,050,646</td>
<td>3,780,182</td>
<td>22,343,802</td>
<td>44,442,573</td>
</tr>
<tr>
<td></td>
<td>16,976,667</td>
<td>4,081,568</td>
<td>4,045,000</td>
<td>21,882,875</td>
<td>44,446,202</td>
</tr>
<tr>
<td>200</td>
<td>26,648,167</td>
<td>8,257,917</td>
<td>3,742,187</td>
<td>33,931,750</td>
<td>89,754,657</td>
</tr>
<tr>
<td></td>
<td>26,139,667</td>
<td>8,398,750</td>
<td>7,512,344</td>
<td>31,273,708</td>
<td>89,326,751</td>
</tr>
<tr>
<td></td>
<td>26,159,792</td>
<td>8,314,596</td>
<td>6,669,948</td>
<td>32,061,396</td>
<td>89,497,619</td>
</tr>
<tr>
<td></td>
<td>25,935,730</td>
<td>8,258,782</td>
<td>7,291,433</td>
<td>31,929,296</td>
<td>89,539,968</td>
</tr>
<tr>
<td></td>
<td>25,711,667</td>
<td>8,202,968</td>
<td>7,912,917</td>
<td>31,791,041</td>
<td>89,582,317</td>
</tr>
<tr>
<td>500</td>
<td>54,403,392</td>
<td>20,211,354</td>
<td>13,700,521</td>
<td>53,889,125</td>
<td>203,971,682</td>
</tr>
<tr>
<td></td>
<td>53,184,166</td>
<td>20,174,323</td>
<td>19,206,823</td>
<td>50,126,500</td>
<td>204,672,591</td>
</tr>
<tr>
<td></td>
<td>53,979,634</td>
<td>19,631,536</td>
<td>16,631,745</td>
<td>54,647,583</td>
<td>203,762,599</td>
</tr>
<tr>
<td></td>
<td>53,581,900</td>
<td>19,907,930</td>
<td>17,919,284</td>
<td>52,387,042</td>
<td>204,217,425</td>
</tr>
<tr>
<td></td>
<td>53,568,887</td>
<td>19,489,824</td>
<td>18,741,536</td>
<td>53,896,541</td>
<td>203,885,130</td>
</tr>
<tr>
<td></td>
<td>53,555,875</td>
<td>19,071,718</td>
<td>19,562,969</td>
<td>55,406,041</td>
<td>203,552,836</td>
</tr>
<tr>
<td>1000</td>
<td>108,850,500</td>
<td>37,591,459</td>
<td>23,935,156</td>
<td>68,158,208</td>
<td>388,586,267</td>
</tr>
<tr>
<td></td>
<td>116,550,000</td>
<td>41,633,334</td>
<td>39,417,865</td>
<td>69,604,500</td>
<td>390,145,280</td>
</tr>
<tr>
<td></td>
<td>103,662,667</td>
<td>39,780,313</td>
<td>31,221,537</td>
<td>68,740,167</td>
<td>388,911,728</td>
</tr>
<tr>
<td></td>
<td>110,106,334</td>
<td>40,706,823</td>
<td>35,319,701</td>
<td>69,172,333</td>
<td>389,528,504</td>
</tr>
<tr>
<td></td>
<td>104,290,584</td>
<td>41,337,995</td>
<td>36,913,809</td>
<td>69,247,229</td>
<td>389,382,847</td>
</tr>
<tr>
<td></td>
<td>98,474,834</td>
<td>41,969,166</td>
<td>38,507,917</td>
<td>69,322,125</td>
<td>389,237,189</td>
</tr>
<tr>
<td>2000</td>
<td>208,684,292</td>
<td>78,184,791</td>
<td>46,466,927</td>
<td>98,425,333</td>
<td>772,082,446</td>
</tr>
<tr>
<td></td>
<td>205,980,959</td>
<td>79,744,062</td>
<td>77,576,406</td>
<td>97,294,041</td>
<td>771,892,650</td>
</tr>
<tr>
<td></td>
<td>200,644,376</td>
<td>79,284,219</td>
<td>59,917,891</td>
<td>98,932,479</td>
<td>772,215,586</td>
</tr>
<tr>
<td></td>
<td>203,312,667</td>
<td>79,514,140</td>
<td>68,747,148</td>
<td>98,113,260</td>
<td>772,054,118</td>
</tr>
<tr>
<td></td>
<td>197,958,563</td>
<td>79,948,893</td>
<td>71,058,001</td>
<td>98,776,443</td>
<td>772,201,422</td>
</tr>
<tr>
<td></td>
<td>192,604,459</td>
<td>80,383,646</td>
<td>73,368,854</td>
<td>99,439,625</td>
<td>772,348,726</td>
</tr>
<tr>
<td>5000</td>
<td>488,628,209</td>
<td>197,851,875</td>
<td>96,636,345</td>
<td>186,073,000</td>
<td>1,889,707,993</td>
</tr>
<tr>
<td></td>
<td>520,830,000</td>
<td>215,698,125</td>
<td>199,148,750</td>
<td>189,826,958</td>
<td>1,902,337,458</td>
</tr>
<tr>
<td></td>
<td>493,344,709</td>
<td>206,088,010</td>
<td>144,844,449</td>
<td>189,428,188</td>
<td>1,886,038,222</td>
</tr>
<tr>
<td></td>
<td>507,087,355</td>
<td>210,893,067</td>
<td>171,996,599</td>
<td>189,627,573</td>
<td>1,894,187,840</td>
</tr>
<tr>
<td></td>
<td>502,574,282</td>
<td>212,608,606</td>
<td>182,524,576</td>
<td>191,205,474</td>
<td>1,888,278,146</td>
</tr>
<tr>
<td>Size (KB)</td>
<td>Galaxy S4</td>
<td>Galaxy S7</td>
<td>OnePlus 3</td>
<td>iPhone 7</td>
<td>iPad 4</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>-----------</td>
<td>------------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>498,061,209</td>
<td>214,324,144</td>
<td>193,052,552</td>
<td>192,783,375</td>
<td>1,882,368,452</td>
</tr>
<tr>
<td>10000</td>
<td>1,014,354,042</td>
<td>415,329,144</td>
<td>176,663,281</td>
<td>305,155,791</td>
<td>3,715,968,876</td>
</tr>
<tr>
<td></td>
<td>973,586,571</td>
<td>397,779,114</td>
<td>378,469,531</td>
<td>300,760,875</td>
<td>3,796,735,699</td>
</tr>
<tr>
<td></td>
<td>993,884,668</td>
<td>393,694,338</td>
<td>278,428,412</td>
<td>306,348,104</td>
<td>3,768,351,145</td>
</tr>
<tr>
<td></td>
<td>983,735,619</td>
<td>395,736,726</td>
<td>328,448,971</td>
<td>303,554,489</td>
<td>3,782,543,422</td>
</tr>
<tr>
<td></td>
<td>973,415,293</td>
<td>372,059,531</td>
<td>380,193,542</td>
<td>307,540,416</td>
<td>3,820,733,415</td>
</tr>
</tbody>
</table>
Table A.8 Execution time of AES-256 decryption on Android and iOS devices in nanoseconds

<table>
<thead>
<tr>
<th>Size (KB)</th>
<th>Galaxy S4</th>
<th>Galaxy S7</th>
<th>OnePlus 3</th>
<th>iPhone 7</th>
<th>iPad 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>134,642,167</td>
<td>83,270,573</td>
<td>64,734,948</td>
<td>26,182,208</td>
<td>2,093,594</td>
</tr>
<tr>
<td></td>
<td>137,461,958</td>
<td>80,967,708</td>
<td>64,945,260</td>
<td>29,516,375</td>
<td>3,289,375</td>
</tr>
<tr>
<td></td>
<td>135,297,980</td>
<td>82,158,516</td>
<td>64,758,958</td>
<td>27,915,083</td>
<td>2,706,302</td>
</tr>
<tr>
<td></td>
<td>136,379,969</td>
<td>81,563,112</td>
<td>64,852,109</td>
<td>28,715,729</td>
<td>2,997,839</td>
</tr>
<tr>
<td></td>
<td>136,166,880</td>
<td>81,304,785</td>
<td>64,817,539</td>
<td>29,181,844</td>
<td>3,158,424</td>
</tr>
<tr>
<td></td>
<td>135,953,792</td>
<td>80,146,458</td>
<td>64,782,968</td>
<td>29,647,958</td>
<td>3,319,010</td>
</tr>
<tr>
<td>200</td>
<td>150,888,750</td>
<td>84,490,156</td>
<td>68,370,000</td>
<td>26,698,458</td>
<td>6,016,302</td>
</tr>
<tr>
<td></td>
<td>161,565,875</td>
<td>84,556,927</td>
<td>68,533,698</td>
<td>27,524,666</td>
<td>5,862,381</td>
</tr>
<tr>
<td></td>
<td>153,911,979</td>
<td>84,500,860</td>
<td>68,427,448</td>
<td>25,892,104</td>
<td>6,130,860</td>
</tr>
<tr>
<td></td>
<td>157,738,927</td>
<td>84,528,893</td>
<td>68,480,573</td>
<td>27,088,385</td>
<td>5,996,620</td>
</tr>
<tr>
<td></td>
<td>157,337,068</td>
<td>84,520,228</td>
<td>68,482,735</td>
<td>25,897,068</td>
<td>6,121,019</td>
</tr>
<tr>
<td></td>
<td>156,935,208</td>
<td>84,511,563</td>
<td>68,484,896</td>
<td>25,085,750</td>
<td>6,245,417</td>
</tr>
<tr>
<td>500</td>
<td>190,067,875</td>
<td>95,264,583</td>
<td>80,076,146</td>
<td>30,647,500</td>
<td>13,418,334</td>
</tr>
<tr>
<td></td>
<td>208,846,542</td>
<td>95,470,364</td>
<td>79,556,614</td>
<td>34,504,916</td>
<td>13,290,260</td>
</tr>
<tr>
<td></td>
<td>193,551,542</td>
<td>95,426,562</td>
<td>79,755,417</td>
<td>33,829,500</td>
<td>13,725,313</td>
</tr>
<tr>
<td></td>
<td>201,199,042</td>
<td>95,448,463</td>
<td>79,656,015</td>
<td>34,167,208</td>
<td>13,507,786</td>
</tr>
<tr>
<td></td>
<td>199,117,125</td>
<td>95,518,502</td>
<td>79,434,687</td>
<td>35,589,354</td>
<td>13,770,039</td>
</tr>
<tr>
<td></td>
<td>197,035,208</td>
<td>95,588,541</td>
<td>79,434,687</td>
<td>37,011,500</td>
<td>14,032,291</td>
</tr>
<tr>
<td>1000</td>
<td>289,997,292</td>
<td>116,741,875</td>
<td>97,601,823</td>
<td>32,055,375</td>
<td>25,832,240</td>
</tr>
<tr>
<td></td>
<td>280,779,063</td>
<td>115,598,081</td>
<td>98,509,219</td>
<td>32,097,958</td>
<td>25,100,079</td>
</tr>
<tr>
<td></td>
<td>314,205,427</td>
<td>115,541,879</td>
<td>99,234,896</td>
<td>34,167,208</td>
<td>25,684,857</td>
</tr>
<tr>
<td></td>
<td>292,883,130</td>
<td>114,998,083</td>
<td>99,325,756</td>
<td>33,829,500</td>
<td>25,026,387</td>
</tr>
<tr>
<td></td>
<td>271,560,833</td>
<td>114,454,287</td>
<td>99,416,615</td>
<td>32,140,541</td>
<td>24,367,917</td>
</tr>
<tr>
<td>2000</td>
<td>375,658,458</td>
<td>153,236,355</td>
<td>137,553,177</td>
<td>38,474,708</td>
<td>54,901,562</td>
</tr>
<tr>
<td></td>
<td>471,803,042</td>
<td>151,440,729</td>
<td>137,230,156</td>
<td>38,674,875</td>
<td>54,433,750</td>
</tr>
<tr>
<td></td>
<td>414,685,021</td>
<td>152,619,948</td>
<td>138,789,844</td>
<td>38,177,625</td>
<td>54,980,469</td>
</tr>
<tr>
<td></td>
<td>443,244,031</td>
<td>152,030,339</td>
<td>138,010,000</td>
<td>38,426,250</td>
<td>54,707,109</td>
</tr>
<tr>
<td></td>
<td>448,477,807</td>
<td>152,016,940</td>
<td>139,018,255</td>
<td>38,153,395</td>
<td>54,883,242</td>
</tr>
<tr>
<td></td>
<td>453,711,583</td>
<td>152,003,541</td>
<td>140,026,510</td>
<td>37,880,541</td>
<td>55,059,375</td>
</tr>
<tr>
<td>5000</td>
<td>745,152,417</td>
<td>280,504,479</td>
<td>257,933,646</td>
<td>44,308,541</td>
<td>96,629,739</td>
</tr>
<tr>
<td></td>
<td>729,416,876</td>
<td>282,966,875</td>
<td>250,835,208</td>
<td>41,128,250</td>
<td>97,110,000</td>
</tr>
<tr>
<td></td>
<td>745,763,125</td>
<td>274,912,865</td>
<td>258,825,209</td>
<td>43,181,458</td>
<td>96,472,760</td>
</tr>
<tr>
<td></td>
<td>737,590,001</td>
<td>278,939,870</td>
<td>254,830,208</td>
<td>42,154,854</td>
<td>96,791,380</td>
</tr>
<tr>
<td></td>
<td>741,981,917</td>
<td>274,130,560</td>
<td>257,273,490</td>
<td>42,104,615</td>
<td>96,553,581</td>
</tr>
<tr>
<td></td>
<td>746,373,833</td>
<td>269,321,250</td>
<td>259,716,771</td>
<td>42,054,375</td>
<td>96,315,781</td>
</tr>
<tr>
<td>Size (KB)</td>
<td>Galaxy S4</td>
<td>Galaxy S7</td>
<td>OnePlus 3</td>
<td>iPhone 7</td>
<td>iPad 4</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>----------------</td>
<td>----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>10000</td>
<td>1,155,961,917</td>
<td>441,889,167</td>
<td>430,471,614</td>
<td>47,128,083</td>
<td>180,958,490</td>
</tr>
<tr>
<td></td>
<td>1,356,368,292</td>
<td>440,065,573</td>
<td>431,309,635</td>
<td>51,090,916</td>
<td>180,810,364</td>
</tr>
<tr>
<td></td>
<td>1,176,304,792</td>
<td>443,960,599</td>
<td>433,624,088</td>
<td>48,952,646</td>
<td>181,285,626</td>
</tr>
<tr>
<td></td>
<td>1,266,336,542</td>
<td>442,013,086</td>
<td>432,466,862</td>
<td>50,021,781</td>
<td>181,047,995</td>
</tr>
<tr>
<td></td>
<td>1,231,492,105</td>
<td>444,022,559</td>
<td>434,621,712</td>
<td>50,399,494</td>
<td>181,330,378</td>
</tr>
<tr>
<td></td>
<td>1,196,647,667</td>
<td>446,032,031</td>
<td>436,776,562</td>
<td>50,777,208</td>
<td>181,612,761</td>
</tr>
</tbody>
</table>
Table A.9 Execution time of RSA-2048 encryption on Android and iOS devices in nanoseconds

<table>
<thead>
<tr>
<th>Size (KB)</th>
<th>Galaxy S4</th>
<th>Galaxy S7</th>
<th>OnePlus 3</th>
<th>iPhone 7</th>
<th>iPad 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>7,687,417</td>
<td>1,094,687</td>
<td>1,060,000</td>
<td>1,068,000</td>
<td>3,617,208</td>
</tr>
<tr>
<td></td>
<td>6,412,250</td>
<td>519,636</td>
<td>1,038,646</td>
<td>1,344,000</td>
<td>3,620,573</td>
</tr>
<tr>
<td></td>
<td>7,125,979</td>
<td>831,406</td>
<td>1,063,073</td>
<td>1,039,208</td>
<td>3,617,985</td>
</tr>
<tr>
<td></td>
<td>6,769,115</td>
<td>675,521</td>
<td>1,050,860</td>
<td>1,191,604</td>
<td>3,619,279</td>
</tr>
<tr>
<td></td>
<td>6,666,828</td>
<td>621,823</td>
<td>1,058,503</td>
<td>1,101,010</td>
<td>3,619,020</td>
</tr>
<tr>
<td></td>
<td>6,564,541</td>
<td>568,125</td>
<td>1,066,146</td>
<td>1,010,416</td>
<td>3,618,761</td>
</tr>
<tr>
<td>192</td>
<td>6,595,334</td>
<td>916,406</td>
<td>1,042,606</td>
<td>1,338,041</td>
<td>3,620,796</td>
</tr>
<tr>
<td></td>
<td>6,546,417</td>
<td>546,510</td>
<td>1,093,490</td>
<td>1,390,708</td>
<td>3,619,779</td>
</tr>
<tr>
<td></td>
<td>6,774,917</td>
<td>714,563</td>
<td>1,052,475</td>
<td>1,176,125</td>
<td>3,621,176</td>
</tr>
<tr>
<td></td>
<td>6,660,667</td>
<td>630,536</td>
<td>1,072,983</td>
<td>1,283,416</td>
<td>3,622,573</td>
</tr>
<tr>
<td></td>
<td>6,807,584</td>
<td>571,628</td>
<td>1,067,663</td>
<td>1,148,812</td>
<td>3,620,660</td>
</tr>
<tr>
<td></td>
<td>6,954,500</td>
<td>512,719</td>
<td>1,062,344</td>
<td>1,014,208</td>
<td>3,621,616</td>
</tr>
<tr>
<td>256</td>
<td>6,399,083</td>
<td>552,240</td>
<td>1,118,282</td>
<td>1,007,833</td>
<td>3,620,880</td>
</tr>
<tr>
<td></td>
<td>6,361,417</td>
<td>523,438</td>
<td>1,301,927</td>
<td>1,019,500</td>
<td>3,620,144</td>
</tr>
<tr>
<td></td>
<td>6,341,229</td>
<td>538,881</td>
<td>1,415,573</td>
<td>1,029,937</td>
<td>3,621,416</td>
</tr>
<tr>
<td></td>
<td>6,351,323</td>
<td>531,159</td>
<td>1,358,750</td>
<td>1,024,719</td>
<td>3,620,780</td>
</tr>
<tr>
<td></td>
<td>6,317,349</td>
<td>528,340</td>
<td>1,535,807</td>
<td>1,038,380</td>
<td>3,621,366</td>
</tr>
<tr>
<td></td>
<td>6,283,375</td>
<td>525,521</td>
<td>1,712,864</td>
<td>1,052,041</td>
<td>3,621,952</td>
</tr>
</tbody>
</table>

158
Table A.10 Execution time of RSA-2048 decryption on Android and iOS devices in nanoseconds

<table>
<thead>
<tr>
<th>Size (KB)</th>
<th>Galaxy S4</th>
<th>Galaxy S7</th>
<th>OnePlus 3</th>
<th>iPhone 7</th>
<th>iPad 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>21,273,961</td>
<td>10,338,333</td>
<td>11,457,813</td>
<td>30,843,750</td>
<td>59,802,166</td>
</tr>
<tr>
<td></td>
<td>20,523,584</td>
<td>10,410,782</td>
<td>11,408,646</td>
<td>34,545,083</td>
<td>59,238,715</td>
</tr>
<tr>
<td></td>
<td>21,124,543</td>
<td>10,359,744</td>
<td>11,376,902</td>
<td>30,604,583</td>
<td>59,799,347</td>
</tr>
<tr>
<td></td>
<td>20,824,063</td>
<td>10,385,263</td>
<td>11,392,774</td>
<td>32,574,833</td>
<td>59,519,031</td>
</tr>
<tr>
<td></td>
<td>20,899,594</td>
<td>10,383,208</td>
<td>11,344,382</td>
<td>31,470,125</td>
<td>59,657,779</td>
</tr>
<tr>
<td></td>
<td>20,975,124</td>
<td>10,381,154</td>
<td>11,295,990</td>
<td>30,365,416</td>
<td>59,796,527</td>
</tr>
<tr>
<td>192</td>
<td>21,594,666</td>
<td>10,268,855</td>
<td>11,375,625</td>
<td>33,842,583</td>
<td>59,801,833</td>
</tr>
<tr>
<td></td>
<td>21,393,792</td>
<td>10,819,948</td>
<td>11,198,178</td>
<td>33,624,791</td>
<td>59,804,812</td>
</tr>
<tr>
<td></td>
<td>21,467,833</td>
<td>10,290,834</td>
<td>11,346,979</td>
<td>33,938,583</td>
<td>59,812,739</td>
</tr>
<tr>
<td></td>
<td>21,430,813</td>
<td>10,555,391</td>
<td>11,272,579</td>
<td>33,781,687</td>
<td>59,808,775</td>
</tr>
<tr>
<td></td>
<td>21,385,906</td>
<td>10,434,102</td>
<td>11,295,456</td>
<td>33,908,135</td>
<td>59,816,210</td>
</tr>
<tr>
<td></td>
<td>21,341,000</td>
<td>10,312,813</td>
<td>11,318,333</td>
<td>34,034,583</td>
<td>59,823,644</td>
</tr>
<tr>
<td>256</td>
<td>21,257,208</td>
<td>10,327,448</td>
<td>11,304,063</td>
<td>33,753,833</td>
<td>59,818,174</td>
</tr>
<tr>
<td></td>
<td>22,003,417</td>
<td>10,401,719</td>
<td>11,346,615</td>
<td>33,939,833</td>
<td>59,810,413</td>
</tr>
<tr>
<td></td>
<td>21,159,354</td>
<td>10,344,167</td>
<td>11,417,969</td>
<td>34,143,104</td>
<td>59,814,402</td>
</tr>
<tr>
<td></td>
<td>21,581,386</td>
<td>10,372,943</td>
<td>11,382,292</td>
<td>34,041,469</td>
<td>59,812,407</td>
</tr>
<tr>
<td></td>
<td>21,321,443</td>
<td>10,366,915</td>
<td>11,457,084</td>
<td>34,286,922</td>
<td>59,811,518</td>
</tr>
<tr>
<td></td>
<td>21,061,500</td>
<td>10,360,886</td>
<td>11,531,875</td>
<td>34,532,375</td>
<td>59,810,629</td>
</tr>
</tbody>
</table>
Table A.11 Execution time of digital signature generation on Android and iOS devices in nanoseconds

<table>
<thead>
<tr>
<th>Size (KB)</th>
<th>Galaxy S4</th>
<th>Galaxy S7</th>
<th>OnePlus 3</th>
<th>iPhone 7</th>
<th>iPad 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>147,140,041</td>
<td>82,087,813</td>
<td>72,297,604</td>
<td>43,868,708</td>
<td>34,549,750</td>
</tr>
<tr>
<td></td>
<td>149,191,625</td>
<td>83,079,792</td>
<td>72,439,791</td>
<td>46,829,750</td>
<td>32,150,000</td>
</tr>
<tr>
<td></td>
<td>146,882,854</td>
<td>82,484,870</td>
<td>72,722,526</td>
<td>42,335,208</td>
<td>32,909,333</td>
</tr>
<tr>
<td></td>
<td>148,037,240</td>
<td>82,782,331</td>
<td>72,581,159</td>
<td>44,582,479</td>
<td>31,899,291</td>
</tr>
<tr>
<td></td>
<td>147,331,453</td>
<td>82,832,129</td>
<td>72,864,303</td>
<td>42,692,094</td>
<td>31,268,916</td>
</tr>
<tr>
<td></td>
<td>146,625,667</td>
<td>82,881,927</td>
<td>73,147,448</td>
<td>40,801,708</td>
<td>31,274,583</td>
</tr>
<tr>
<td>200</td>
<td>163,011,167</td>
<td>82,929,114</td>
<td>72,671,302</td>
<td>42,793,666</td>
<td>31,274,583</td>
</tr>
<tr>
<td></td>
<td>156,206,500</td>
<td>83,622,969</td>
<td>72,798,021</td>
<td>41,248,500</td>
<td>30,942,093</td>
</tr>
<tr>
<td></td>
<td>155,162,959</td>
<td>83,195,260</td>
<td>72,961,355</td>
<td>40,801,708</td>
<td>30,186,416</td>
</tr>
<tr>
<td></td>
<td>155,684,729</td>
<td>83,409,115</td>
<td>73,679,688</td>
<td>41,792,719</td>
<td>30,529,083</td>
</tr>
<tr>
<td></td>
<td>151,499,740</td>
<td>83,435,260</td>
<td>73,065,547</td>
<td>42,336,937</td>
<td>30,842,093</td>
</tr>
<tr>
<td></td>
<td>147,314,750</td>
<td>83,461,406</td>
<td>73,251,407</td>
<td>41,880,208</td>
<td>30,942,093</td>
</tr>
<tr>
<td>500</td>
<td>176,424,500</td>
<td>83,900,365</td>
<td>75,847,240</td>
<td>45,843,416</td>
<td>30,204,666</td>
</tr>
<tr>
<td></td>
<td>167,991,750</td>
<td>84,246,042</td>
<td>73,683,490</td>
<td>45,977,166</td>
<td>31,175,791</td>
</tr>
<tr>
<td></td>
<td>189,207,563</td>
<td>84,211,901</td>
<td>75,102,813</td>
<td>41,978,208</td>
<td>30,942,093</td>
</tr>
<tr>
<td></td>
<td>178,599,656</td>
<td>84,228,972</td>
<td>74,393,151</td>
<td>43,977,687</td>
<td>31,354,417</td>
</tr>
<tr>
<td></td>
<td>190,295,141</td>
<td>84,376,204</td>
<td>74,375,768</td>
<td>43,354,177</td>
<td>30,942,093</td>
</tr>
<tr>
<td></td>
<td>201,990,625</td>
<td>84,523,437</td>
<td>74,358,385</td>
<td>42,730,666</td>
<td>30,529,083</td>
</tr>
<tr>
<td>1000</td>
<td>200,594,333</td>
<td>85,731,979</td>
<td>76,219,011</td>
<td>45,843,416</td>
<td>30,204,666</td>
</tr>
<tr>
<td></td>
<td>212,944,958</td>
<td>86,261,667</td>
<td>76,049,479</td>
<td>45,255,375</td>
<td>32,757,791</td>
</tr>
<tr>
<td></td>
<td>207,200,521</td>
<td>86,002,709</td>
<td>76,108,099</td>
<td>45,768,896</td>
<td>31,863,437</td>
</tr>
<tr>
<td></td>
<td>210,072,740</td>
<td>86,132,188</td>
<td>76,078,789</td>
<td>45,512,135</td>
<td>32,310,614</td>
</tr>
<tr>
<td></td>
<td>211,939,724</td>
<td>86,202,813</td>
<td>76,037,988</td>
<td>45,603,255</td>
<td>32,916,411</td>
</tr>
<tr>
<td></td>
<td>213,806,709</td>
<td>86,273,438</td>
<td>75,997,187</td>
<td>45,694,375</td>
<td>33,522,208</td>
</tr>
<tr>
<td>2000</td>
<td>222,490,000</td>
<td>86,542,031</td>
<td>78,828,021</td>
<td>52,923,208</td>
<td>32,032,333</td>
</tr>
<tr>
<td></td>
<td>290,238,959</td>
<td>90,085,364</td>
<td>79,191,406</td>
<td>55,773,791</td>
<td>30,186,416</td>
</tr>
<tr>
<td></td>
<td>225,909,084</td>
<td>88,067,422</td>
<td>79,300,651</td>
<td>52,843,229</td>
<td>31,881,667</td>
</tr>
<tr>
<td></td>
<td>258,074,021</td>
<td>89,076,393</td>
<td>79,246,029</td>
<td>54,308,510</td>
<td>31,034,041</td>
</tr>
<tr>
<td></td>
<td>243,701,094</td>
<td>89,334,603</td>
<td>79,509,655</td>
<td>53,535,880</td>
<td>31,382,521</td>
</tr>
<tr>
<td></td>
<td>229,328,167</td>
<td>89,592,813</td>
<td>79,773,281</td>
<td>52,763,250</td>
<td>31,731,000</td>
</tr>
<tr>
<td>5000</td>
<td>224,751,250</td>
<td>87,652,656</td>
<td>86,292,010</td>
<td>47,441,750</td>
<td>26,664,375</td>
</tr>
<tr>
<td></td>
<td>301,015,625</td>
<td>92,883,802</td>
<td>87,628,334</td>
<td>54,234,208</td>
<td>27,561,291</td>
</tr>
<tr>
<td></td>
<td>229,120,042</td>
<td>89,098,281</td>
<td>84,218,219</td>
<td>48,804,104</td>
<td>26,593,688</td>
</tr>
<tr>
<td></td>
<td>265,067,834</td>
<td>90,991,042</td>
<td>85,923,276</td>
<td>51,212,156</td>
<td>26,550,489</td>
</tr>
<tr>
<td></td>
<td>249,278,334</td>
<td>90,767,474</td>
<td>84,033,852</td>
<td>50,075,307</td>
<td>25,482,745</td>
</tr>
<tr>
<td></td>
<td>233,488,834</td>
<td>90,543,906</td>
<td>82,144,427</td>
<td>48,938,458</td>
<td>24,415,000</td>
</tr>
<tr>
<td>Size (KB)</td>
<td>Galaxy S4</td>
<td>Galaxy S7</td>
<td>OnePlus 3</td>
<td>iPhone 7</td>
<td>iPad 4</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>-------------</td>
<td>------------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>10000</td>
<td>413,551,167</td>
<td>93,601,459</td>
<td>86,773,125</td>
<td>52,250,000</td>
<td>21,255,833</td>
</tr>
<tr>
<td></td>
<td>387,704,625</td>
<td>98,254,219</td>
<td>91,073,434</td>
<td>48,719,208</td>
<td>21,477,208</td>
</tr>
<tr>
<td></td>
<td>384,172,063</td>
<td>95,761,641</td>
<td>87,186,016</td>
<td>51,889,625</td>
<td>21,029,104</td>
</tr>
<tr>
<td></td>
<td>385,938,344</td>
<td>97,007,930</td>
<td>89,129,725</td>
<td>50,304,417</td>
<td>21,253,156</td>
</tr>
<tr>
<td></td>
<td>370,365,652</td>
<td>97,464,877</td>
<td>88,364,315</td>
<td>50,916,833</td>
<td>21,027,766</td>
</tr>
<tr>
<td></td>
<td>354,792,959</td>
<td>97,921,823</td>
<td>87,598,906</td>
<td>51,529,250</td>
<td>20,802,375</td>
</tr>
</tbody>
</table>
Table A.12 Execution time of digital signature verification on Android and iOS devices in nanoseconds

<table>
<thead>
<tr>
<th>Size (KB)</th>
<th>Galaxy S4</th>
<th>Galaxy S7</th>
<th>OnePlus 3</th>
<th>iPhone 7</th>
<th>iPad 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>128,602,917</td>
<td>72,455,677</td>
<td>62,856,927</td>
<td>11,961,000</td>
<td>1,341,541</td>
</tr>
<tr>
<td></td>
<td>131,817,750</td>
<td>72,459,791</td>
<td>62,628,437</td>
<td>11,769,208</td>
<td>1,038,250</td>
</tr>
<tr>
<td></td>
<td>128,321,332</td>
<td>72,072,761</td>
<td>62,856,615</td>
<td>11,960,792</td>
<td>1,213,979</td>
</tr>
<tr>
<td></td>
<td>130,069,541</td>
<td>72,266,276</td>
<td>62,742,526</td>
<td>11,865,000</td>
<td>1,126,114</td>
</tr>
<tr>
<td></td>
<td>129,054,643</td>
<td>71,978,060</td>
<td>62,799,414</td>
<td>11,912,791</td>
<td>1,106,265</td>
</tr>
<tr>
<td></td>
<td>128,039,746</td>
<td>71,689,844</td>
<td>62,856,302</td>
<td>11,960,583</td>
<td>1,086,416</td>
</tr>
<tr>
<td>200</td>
<td>140,646,708</td>
<td>73,066,979</td>
<td>63,694,375</td>
<td>11,955,541</td>
<td>1,330,458</td>
</tr>
<tr>
<td></td>
<td>138,822,083</td>
<td>72,737,552</td>
<td>63,786,510</td>
<td>12,571,416</td>
<td>1,125,666</td>
</tr>
<tr>
<td></td>
<td>140,179,896</td>
<td>72,792,162</td>
<td>63,634,740</td>
<td>11,913,812</td>
<td>1,196,750</td>
</tr>
<tr>
<td></td>
<td>139,500,989</td>
<td>72,641,100</td>
<td>63,426,865</td>
<td>12,057,349</td>
<td>1,161,208</td>
</tr>
<tr>
<td></td>
<td>139,713,083</td>
<td>72,517,344</td>
<td>63,575,105</td>
<td>11,872,083</td>
<td>1,063,041</td>
</tr>
<tr>
<td>500</td>
<td>140,800,333</td>
<td>73,683,907</td>
<td>64,445,052</td>
<td>14,159,625</td>
<td>1,306,458</td>
</tr>
<tr>
<td></td>
<td>140,540,375</td>
<td>73,635,468</td>
<td>64,090,885</td>
<td>14,894,250</td>
<td>1,098,208</td>
</tr>
<tr>
<td></td>
<td>144,689,563</td>
<td>73,758,047</td>
<td>64,412,448</td>
<td>14,118,938</td>
<td>1,172,146</td>
</tr>
<tr>
<td></td>
<td>142,614,969</td>
<td>73,696,758</td>
<td>64,251,667</td>
<td>14,506,594</td>
<td>1,135,177</td>
</tr>
<tr>
<td></td>
<td>145,596,880</td>
<td>73,764,472</td>
<td>64,315,755</td>
<td>14,292,422</td>
<td>1,086,505</td>
</tr>
<tr>
<td></td>
<td>148,578,792</td>
<td>73,832,187</td>
<td>64,379,844</td>
<td>14,078,250</td>
<td>1,063,833</td>
</tr>
<tr>
<td>1000</td>
<td>212,352,584</td>
<td>73,295,938</td>
<td>65,932,395</td>
<td>14,238,916</td>
<td>1,087,125</td>
</tr>
<tr>
<td></td>
<td>217,680,083</td>
<td>75,648,125</td>
<td>66,552,552</td>
<td>14,954,875</td>
<td>1,046,291</td>
</tr>
<tr>
<td></td>
<td>209,752,896</td>
<td>74,833,829</td>
<td>66,645,364</td>
<td>14,516,708</td>
<td>1,064,083</td>
</tr>
<tr>
<td></td>
<td>190,716,490</td>
<td>75,240,977</td>
<td>66,598,958</td>
<td>14,735,792</td>
<td>1,055,187</td>
</tr>
<tr>
<td></td>
<td>198,934,849</td>
<td>75,806,348</td>
<td>66,978,646</td>
<td>14,765,146</td>
<td>1,048,114</td>
</tr>
<tr>
<td></td>
<td>207,153,208</td>
<td>76,371,719</td>
<td>67,358,333</td>
<td>14,794,500</td>
<td>1,041,041</td>
</tr>
<tr>
<td>2000</td>
<td>272,120,792</td>
<td>79,846,927</td>
<td>68,727,657</td>
<td>20,118,916</td>
<td>1,100,916</td>
</tr>
<tr>
<td></td>
<td>198,318,458</td>
<td>78,381,875</td>
<td>69,031,042</td>
<td>20,140,666</td>
<td>1,042,208</td>
</tr>
<tr>
<td></td>
<td>257,873,646</td>
<td>79,246,979</td>
<td>68,460,573</td>
<td>20,276,396</td>
<td>1,078,479</td>
</tr>
<tr>
<td></td>
<td>228,096,052</td>
<td>78,814,427</td>
<td>68,745,808</td>
<td>20,208,531</td>
<td>1,060,343</td>
</tr>
<tr>
<td></td>
<td>235,861,276</td>
<td>78,730,729</td>
<td>68,469,648</td>
<td>20,321,203</td>
<td>1,058,192</td>
</tr>
<tr>
<td></td>
<td>243,626,500</td>
<td>78,647,031</td>
<td>68,193,489</td>
<td>20,433,875</td>
<td>1,056,041</td>
</tr>
<tr>
<td>5000</td>
<td>282,697,917</td>
<td>83,933,333</td>
<td>76,462,552</td>
<td>23,519,750</td>
<td>1,105,250</td>
</tr>
<tr>
<td></td>
<td>266,575,793</td>
<td>79,540,833</td>
<td>76,739,427</td>
<td>23,815,208</td>
<td>1,087,416</td>
</tr>
<tr>
<td></td>
<td>264,419,834</td>
<td>82,956,901</td>
<td>76,067,735</td>
<td>23,535,417</td>
<td>1,133,500</td>
</tr>
<tr>
<td>Size (KB)</td>
<td>Galaxy S4</td>
<td>Galaxy S7</td>
<td>OnePlus 3</td>
<td>iPhone 7</td>
<td>iPad 4</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>-----------</td>
<td>------------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>265,497,814</td>
<td>81,248,867</td>
<td>76,403,581</td>
<td>23,675,312</td>
<td>1,110,458</td>
<td></td>
</tr>
<tr>
<td>255,819,782</td>
<td>81,614,667</td>
<td>76,038,249</td>
<td>23,613,198</td>
<td>1,136,104</td>
<td></td>
</tr>
<tr>
<td>246,141,751</td>
<td>81,980,468</td>
<td>75,672,917</td>
<td>23,551,083</td>
<td>1,161,750</td>
<td></td>
</tr>
<tr>
<td>322,933,709</td>
<td>86,216,041</td>
<td>79,325,886</td>
<td>29,072,250</td>
<td>1,070,958</td>
<td></td>
</tr>
<tr>
<td>339,760,208</td>
<td>85,661,563</td>
<td>80,467,916</td>
<td>29,179,125</td>
<td>1,107,041</td>
<td></td>
</tr>
<tr>
<td>336,879,480</td>
<td>87,687,630</td>
<td>77,663,907</td>
<td>29,427,292</td>
<td>1,085,104</td>
<td></td>
</tr>
<tr>
<td>338,319,844</td>
<td>86,674,597</td>
<td>79,065,911</td>
<td>29,303,208</td>
<td>1,096,073</td>
<td></td>
</tr>
<tr>
<td>344,572,547</td>
<td>87,916,908</td>
<td>77,533,919</td>
<td>29,542,771</td>
<td>1,097,661</td>
<td></td>
</tr>
<tr>
<td>350,825,250</td>
<td>89,159,219</td>
<td>76,001,927</td>
<td>29,782,333</td>
<td>1,099,250</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX B

PERFORMANCE MEASUREMENT TOOL

In this Appendix, we provide a brief description and screenshots of the performance measurement tools developed based on the HALA framework. The purpose of these tools is to measure and collect the execution times of several cryptographic methods on mobile devices. Two versions have been developed for Android and iOS mobile devices. Both version have the same functions and use the same test configurations. The tools provide easy functions to generate and store the needed encryption keys on the devices.

CryptoTest, a simple implementation of the HALA framework for iOS devices, is developed to measure the execution times for some cryptographic primitives. A revised version of this tool will be made available for Android devices on a later date. Please check the following link later to find the latest version of the Android tool:

http://cs.uccs.edu/~gsc/pub/phd/nalamri/src/

The Xcode project can be downloaded from the following link:

http://cs.uccs.edu/~gsc/pub/phd/nalamri/src/CryptoTest.zip

Installation Requirements:

1. A recent version of the Xcode IDE.

2. An iOS device running iOS 9.3 or later. Otherwise, the tool can be tested on the Xcode simulation tool.

Installation Steps:

1. Download the .zip file of the CryptoTest project on a macOS device from the aforementioned link.

2. Unzip the file in your location of choice.

3. Open the project in Xcode.

4. Connect the iOS device to the macOS device and run the project. Alternatively, the project can be built and run on a simulator.

5. If running the CryptoTest app for the first time, Xcode will request the verification of the developer certificate. On the iOS device, go to Settings ➔ General ➔ Device Management. Trust apps form developer nayf_med@hotmail.com.

6. Once the application is installed, it will automatically generate the needed encryption keys and test files and the app will be ready to use.

In this section, we show some examples from the tool running on iPhone 8 Plus.
Figure B.1 The first screen in the application provides the ability to run all the cryptographic methods at once. These methods are run independently and consecutively and the result from the run of each method is not affected by the others.
Figure B.2 In this screen, the user can generate secure random data by choosing needed data size from the list (8, 16, 32, 64, 128, and 256 bytes). The execution-times from 20 consecutive runs are displayed in nanoseconds. The average time is displayed in the upper box.
Figure B.3 In this screen, the user can generate and verify hash codes using the SHA-2 hash function. Different data sizes can be added easily to the code. In this implementation, we used a 64-byte test file. The user can choose whether to generate a new hash code from a file or to verify a hash code received with another file. The execution-times from 20 consecutive runs are displayed in nanoseconds. The average time is displayed in the lower box.
Figure B.4 In this screen, the user can encrypt or decrypt existing files using AES-256. Different data sizes can be added easily to the code. In this implementation, we used a 64-byte test file. The user can choose to run encryption or decryption. The execution-times from 20 consecutive runs are displayed in nanoseconds. The average time is displayed in the lower box.
Figure B.5 In this screen, the user can encrypt or decrypt existing files using RSA-2048. Different data sizes can be added easily to the code. In this implementation, we used a 64-byte test file. The user can choose to run encryption or decryption. The execution-times from 20 consecutive runs are displayed in nanoseconds. The average time is displayed in the lower box.
Figure B.6 In this screen, the user can generate or verify digital signatures. The digital signature in this implementation is a signed hash code generated using SHA-2 and RSA-2048. Different data sizes can be added easily to the code. In this implementation, we used a 64-byte test file. The user can choose to run digital signature generation or verification. The execution-times from 20 consecutive runs are displayed in nanoseconds. The average time is displayed in the lower box.