NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

MASTER STUDIES
TELECOMMUNICATION SYSTEMS AND INTERNET TECHOLOGIES

MSc THESIS

Secure Geo-location Techniques using Trusted Hyper-visor

Savvas G. Rostantis

Supervisors **Eustathios Hadjiefthymiades,** Professor
Anestis Papakotoulas, PhD student

ATHENS

JANUARY 2020

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΔΙΚΤΥΑΚΕΣ ΤΕΧΝΟΛΟΓΙΕΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ασφαλείς τεχνικές υπολογισμού γεωγραφικής θέσης χρησιμοποιώντας Hyper-visor.

Σάββας Γ. Ροστάντης

Επιβλέποντες: **Ευστάθιος Χατζηευθυμιάδης,** Καθηγητής
Ανέστης Παπακοτούλας, Υποψήφιος Διδάκτωρ

ΑΘΗΝΑ

ΙΑΝΟΥΑΡΙΟΣ 2020

MSc THESIS

Secure Geo-location Techniques using Trusted Hyper-visor

Savvas G. Rostantis

S.N.: M1470

SUPERVISOR: **Eustathios Hadjiefthymiades,** Professor
Anestis Papakotoulas, PhD student

January 2020

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ασφαλείς τεχνικές υπολογισμού γεωγραφικής θέσης χρησιμοποιώντας Hyper-visor

Σάββας Γ. Ροστάντης

Α.Μ.: M1470

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Ευστάθιος Χατζηευθυμιάδης,** Καθηγητής
Ανέστης Παπακοτούλας, Υποψήφιος Διδάκτωρ

Ιανουάριος 2020

ABSTRACT

For many, geo-location is a simple process where with the utilization of GPS a person can be located wherever and whenever is requested. However, even if the utilization of GPS for geolocation is the most common way and accurate as a system, it is a huge consumption of energy in order to achieve this process and it lucks on safety mechanisms and techniques. The purpose of this paper is to present another view of how we could locate an unknown node position in a system and how a safe environment could be created for this node. Our main idea was about the creation of a framework where we could create a three-dimensional field in which an unknown node could be located and afterwards a safe environment would be created for the new node. After a research on papers relevant with three-dimensional geo-localization mechanisms and techniques, alongside with the concept of hypervisors for the creation of safe environment with the utilization of cryptography, we came to the conclusion of the creation of a framework which would satisfy those requirements. We created a 3-Dimentional field of four base nodes stations, where we utilized two localization GPS-free algorithms for the location of a fifth unknown node alongside with a hypervisor for the trust environment creation. We utilized a TPM for the cryptography mechanisms and safety keys creation. In this paper we created a simulation where we compare the performance of those two geolocation algorithms in terms of accuracy and computation speed and accuracy, alongside with the hypervisor's security mechanisms performance and its ability for data integrity insurance. Except our proposed framework components, we present also further information that we found in relevant papers, such as a variety of hypervisors and a variety of localization techniques, for more information for future work alongside with implementation steps and guidance.

SUBJECT AREA: Geolocation and System's Security

KEYWORDS: GPS, Hypervisors, TPM, Geolocation, Security, Cryptography, Triangulation, Multilateration, Tetrahedron, Localization, Trust

ΠΕΡΙΛΗΨΗ

Για πολλούς, η γεωγραφική θέση είναι μια απλή διαδικασία όπου με τη χρήση του GPS ένα άτομο μπορεί να εντοπιστεί όπου και όποτε ζητείται. Ωστόσο, ακόμη και αν η χρήση του GPS για γεωγραφική τοποθέτηση είναι ο πιο συνηθισμένος τρόπος και ταυτόχρονα ακριβής ως σύστημα, αποτελεί μια τεράστια κατανάλωση ενέργειας για να επιτευχθεί αυτή η διαδικασία και υστερεί σε μηχανισμούς και τεχνικές ασφαλείας. Σκοπός αυτής της εργασίας είναι να παρουσιάσουμε μια άλλη όψη για το πώς μπορούμε να εντοπίσουμε μια άγνωστη θέση ενός κόμβου σε ένα σύστημα και πώς θα μπορούσε να δημιουργηθεί ένα ασφαλές περιβάλλον για αυτόν τον κόμβο. Βασική μας ιδέα ήταν η δημιουργία ενός μηχανισμού όπου θα μπορούσαμε να δημιουργήσουμε ένα τρισδιάστατο πεδίο στο οποίο θα μπορούσε να εντοπιστεί άγνωστος κόμβος και στη συνέχεια θα δημιουργηθεί ένα ασφαλές περιβάλλον για τον νέο κόμβο. Μετά από μια έρευνα σε δημοσιεύσεις σχετικά με τρισδιάστατους μηχανισμούς και τεχνικές γεω-εντοπισμού, παράλληλα με την έννοια των hypervisors για τη δημιουργία ασφαλούς περιβάλλοντος με την αξιοποίηση της κρυπτογραφίας, καταλήξαμε στο συμπέρασμα της δημιουργίας ενός πλαισίου που θα ικανοποιούσε αυτά απαιτήσεις. Δημιουργήσαμε ένα τρισδιάστατο πεδίο τεσσάρων σταθμών κόμβων, όπου χρησιμοποιήσαμε δύο αλγορίθμους εντοπισμού, χωρίς GPS, για τον εντοπισμό της θέση ενός πέμπτου άγνωστου κόμβου παράλληλα με έναν hypervisor για τη δημιουργία περιβάλλοντος εμπιστοσύνης. Χρησιμοποιήσαμε ένα TPM για τη δημιουργία κρυπτογραφικών μηχανισμών και κλειδιών ασφαλείας. Σε αυτή την εργασία δημιουργήσαμε μια προσομοίωση όπου συγκρίνουμε την απόδοση αυτών των δύο αλγορίθμων γεωγραφικής τοποθέτησης από την άποψη της ταχύτητας και της ακρίβειας του υπολογισμού, παράλληλα με την απόδοση των μηχανισμών ασφαλείας του hypervisor και την ικανότητά του για ασφάλιση ακεραιότητας δεδομένων. Εκτός από τα συστατικά του προτεινόμενου μηχανισμού, παρουσιάζουμε και άλλες πληροφορίες που βρήκαμε σε σχετικά έγγραφα, όπως μια ποικιλία από hypervisors και μια ποικιλία τεχνικών εντοπισμού, για περισσότερες πληροφορίες για μελλοντικές εργασίες παράλληλα με τα βήματα υλοποίησης και εκτέλεσης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Γεο-Εντοπισμός και ασφάλεια συστημάτων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: GPS, Hypervisors, TPM, Γεω-τοποθέτηση, Ασφάλεια, Κρυπτογραφία, Τριγωνισμός, Multilateration, Τετράεδρο, Εντοπισμός, Εμπιστοσύνη

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor doctor Anestis Papakotoulas and professor Eustathios Xatzieuthimiades of the department of Informatics and Telecommunications at National and Kapodistrian University of Athens. The door to Prof. Xatzieuthimiades and PhD student. Papakotoulas office was always open whenever I ran into a trouble spot or had a question about my research or writing. They consistently allowed this paper to be my own work but steered me in the right the direction whenever they thought I needed it. I would also like to thank the experts who were involved in the validation survey for this research project. Without their passionate participation and input, the validation survey could not have been successfully conducted. Finally, I must express my very profound gratitude to my parents and to my friends and girlfriend for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Author

Savvas Rostantis

CONTENTS

ABSTRACT .... **I**

ACKNOWLEDGEMENTS .... **III**

LIST OF FIGURES .... **VII**

LIST OF TABLES .... **X**

1. INTRODUCTION **1**

1.1 Motivation and objective **1**
1.2 Research questions **2**
1.3 Thesis structure and reading guide **3**
1.4 Thesis concept **4**

2. INTRODUCING GEO-LOCATION **5**

2.1 Overview **5**
2.2 Why we need Geo-location? **6**
2.1.1 The History of Geo-location *7*
2.1.2 Applications *9*

3. INTRODUCING LOCALIZATION TECHNIQUES **12**

3.1 Localization techniques classification **12**
3.1.1 GPS Based Localization *12*
3.1.2 GPS Free Localization *13*
3.1.3 Range Base Positioning techniques *15*
3.1.4 Range Free Positioning techniques *18*
3.2 Localization Algorithms **21**
3.2.1 FP-MPP-APIT *21*
3.2.2 3D-IDCP *22*
3.2.3 Novel Centroid *22*
3.2.4 DFPLE *23*
3.2.5 CPE *25*
3.2.6 3D DV-Hop *26*
3.2.7 Enhanced APIT algorithm *28*
3.2.8 Hybrid 3D Localization Algorithm *29*
3.2.9 3D-TDOA Fictitious Point Method *30*
3.2.10 3D-TDOA CHAN Method *33*
3.2.11 TOA/RSSI Direct Location Method *34*
3.2.12 Hybrid 3D-TOA/TDOA *34*
3.2.13 3D-RSSI/TOA Multilateration *34*
3.2.14 Localization Algorithms Summary *35*

4. INTRODUCING SECURITY IN GEO-LOCATION **36**

4.1 Attacks in Localization **36**
4.1.1 Attacks on Nodes *37*
4.1.2 Attacks on Information *39*
4.1.3 Denial of Service Attacks (DoS) *40*
4.1.4 Localization attacks summary *42*
4.2 Secure Geo-location Schemes **42**
4.2.1 Location-Dependent data Encryption Algorithm (LDEA) *43*

4.2.2 Mutual Authentication Insider Node Validation 44
4.2.3 TPM Based Geo-location 46
4.2.4 Authenticated Location based on DRM 47
4.2.5 TOA-ECC Elliptic Curve Cryptography 49
4.2.6 Collaborative localization based on Trust model 50
4.2.7 Secure DV-Hop Localization algorithm 51
4.4 Conclusion 52

5. HYPERVISORS AND GEOLOCATION 55

5.1 Hypervisors Overview 55
5.2 Trust Platform Module 57
5.2.1 Introduction on Trust Platform Module 57
5.2.2 TPM implementations 58
5.3 Trusted Hypervisors 60
5.3.1 XMHF- uberXMHF 60
5.3.2 Xvisor 65
5.3.3 TGVisor 68
5.3.4 TrustVisor 72
5.3.5 SecVisor 77
5.3.6 Lockdown 80
5.3.7 Credo 84
5.3.8 Hypervisors Summary 88

6. PROPOSED SECURE GEO-LOCALIZATION FRAMEWORK 89

6.1 Introduction 89
6.2 Framework Overview 91
6.2.1 Trust Field Creation 91
6.2.2 Node Position Calculation 92
6.2.3 Verification 94
6.2.4 Node Access 95
6.3 Framework Implementation Overview 96
6.3.1 Simulation Overview 96
6.3.2 Random Movement 96
6.3.3 Calculate Metrics 98
6.3.4 Calculate Position 98

7. PROPOSED FRAMEWORK SIMULATION 103

7.1 Framework simulation overview 103
7.1.1 Localization algorithms simulation process 103
7.1.2 Hypervisor simulation process 104
7.1.3 Framework Folder Structure 104
7.1.4 Framework Implementation 105
7.2 Framework Requirements 107
7.2.1 Localization simulation 107
7.2.2 Hypervisor simulation 107
7.3 Framework Execution 107
7.4 Simulation input data 108
7.4.1 Field input 108
7.4.2 Nodes coordinates input 108
7.4.3 Brownian motion input 108
7.4.4 Extra parameters input 108

7.5 Simulation output data **108**
7.5.1 Graphics 108
7.5.2 Statistics files 110
7.5.3 Summary files 110

8. VALIDATION **111**

8.1 Localization algorithms accuracy performance validation **111**
8.1.1 3D-RSSI Multilateration 111
8.1.2 3D-TOA Multilateration 112
8.1.3 3D-RSSI/TOA Direct method 113
8.1.4 Hybrid 3D-TOA/TDOA 115
8.1.5 Chan method 116
8.1.7 Performance Summary 117
8.2 Localization algorithms time validation **118**
8.2.1 3D-RSSI Multilateration 118
8.2.2 3D-TOA Multilateration 118
8.2.3 3D-RSSI Direct method 118
8.2.4 3D-TOA Direct method 118
8.2.5 Hybrid 3D-TOA/TDOA 119
8.2.6 Chan method 119
8.2.7 Performance Summary 120
8.3 Summary **121**

9. CONCLUSION **122**

10. FUTURE WORK **123**

APPENDIX A **124**

Hypervisors Implementation **124**

APPENDIX B **134**

Sequencing in Three-Dimensional Grids python **134**

APPENDIX C **136**

Intersection points between 3 spheres python **136**

APPENDIX D **137**

RANDOM WALK MOBILITY ALGORITHM R **137**

APPENDIX E **138**

Way points output example **138**

ACRONYMS **139**

REFERENCES **141**

LIST OF FIGURES

Figure 1. GPS Battery consumption. 2
Figure 2. Proposed framework concept. 4
Figure 3. Thesis concept 4
Figure 4. GPS Geo-location 5
Figure 5. GPS Market Size, from Grand View Research 7
Figure 6. GPS Trilateration Model 13
Figure 7 .Classification of localization techniques 13
Figure 8. Range Based Vs Range Free 14
Figure 9. RSSI range performance 15
Figure 10. TOA Overview 16
Figure 11. TOA Trilateration 16
Figure 12. TDOA Trilateration 17
Figure 13. AOA Trilateration 18
Figure 14. The APIT principle 20
Figure 15. Fermat Point in Triangle 21
Figure 16. Mid-Perpendicular plane model 21
Figure 17. DFPLE Operation 25
Figure 18. CPE Algorithm 25
Figure 19. Improved CPE algorithm 26
Figure 20. PIT diagram 28
Figure 21. 3D localization algorithm based on APIT and DV-Hop 29
Figure 22. The invisible node attacks 37
Figure 23. The stolen identity attacks 38
Figure 24. Wormhole attack 39
Figure 25. Selective forwarding 40
Figure 26. LDEA Process 43
Figure 27. Propagation time estimation process 46
Figure 28. TPM Based Localization Framework 46
Figure 29. CSLT five sub-processes 51
Figure 30. Hypervisor Overview 55
Figure 31. Hypervisor types 56
Figure 32. Trust Platform Module overview 57
Figure 33. XMHF platform architecture 61
Figure 34. Porting status of several HyperVisors. 63
Figure 35. XMHF Application Benchmarks 63
Figure 36. XMHF performance comparison with Xen 64
Figure 37. XVisor Architecture Overview 66
Figure 38. TGVisor architecture 70
Figure 39. TGVisor Attestation Protocol 71
Figure 40. TGVisor comparison with other HyperVisors 72
Figure 41. TGVisor Javascript performance 72
Figure 42. TrustVisor Architecture Overview 73

Figure 43. TrustVisor mode types 73
Figure 44. Comparison between TrustVisor and Linux Native. 77
Figure 45. SecVisor VS Xen performance.. 80
Figure 46. Lockdown Overview 80
Figure 47. Lockdown protection mechanism 82
Figure 48. Credo comparison with Hyper-V 85
Figure 49. Credo Performance Evaluation CPU/Memory and Disk 87
Figure 50. Proposed framework high level overview. 90
Figure 51. Framework overview 91
Figure 52. Trust field representation.. 92
Figure 53. Intersection point calculation 93
Figure 54. Verification and access overview. 95
Figure 55. Simulation Movement steps 96
Figure 56. Unknown node movement.. 96
Figure 57. Geolocation simulation process 103
Figure 58. Hypervisor simulation process 104
Figure 59. Simulation folder structure.. 105
Figure 60. Python Simulator overview.. 105
Figure 61. Base stations communication process 106
Figure 62. Base communication architecture 107
Figure 63. TDOA, TOA and RSSI simulated values 108
Figure 64. X, Y, Z performance from algorithms 109
Figure 65. time performance algorithm.. 109
Figure 66. Unknown node motion 109
Figure 67. Statistics files structure.. 110
Figure 68. Summary files structure.. 110
Figure 69. X, Y, Z RSSI Mult Alg, PC-Intel_Core_i3-1.7GHz_RAM4GB 111
Figure 70. X, Y, Z RSSI Mult Alg, PC-Intel_Core_i7-2.7GHz_RAM8GB 111
Figure 71. X, Y, Z RSSI Mult Alg, PC-Intel_Core_i7-2.7GHz_RAM16GB 112
Figure 72. X, Y, Z RSSI Mult Alg, Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB....... 112
Figure 73. X, Y, Z TOA Mult Alg, PC-Intel_Core_i3-1.7GHz_RAM4GB 112
Figure 74. X, Y, Z TOA Mult Alg, PC-Intel_Core_i7-2.7GHz_RAM8GB 113
Figure 75. X, Y, Z TOA Mult Alg, PC-Intel_Core_i7-2.7GHz_RAM16GB 113
Figure 76. X, Y, Z TOA Mult Alg, Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB........ 113
Figure 77. X, Y, Z RSSI Direct Alg, PC-Intel_Core_i3-1.7GHz_RAM4GB.................. 114
Figure 78. X, Y, Z RSSI Direct Alg, PC-Intel_Core_i7-2.7GHz_RAM8GB.................. 114
Figure 79. X, Y, Z RSSI Direct Alg, PC-Intel_Core_i7-2.7GHz_RAM16GB................. 114
Figure 80. X, Y, Z RSSI Direct Alg, Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB 114
Figure 81. X, Y, Z Hybrid 3D-TOA/TDOA, PC-Intel_Core_i3-1.7GHz_RAM4GB 115
Figure 82. X, Y, Z Hybrid 3D-TOA/TDOA, PC-Intel_Core_i7-2.7GHz_RAM8GB 115
Figure 83. X, Y, Z Hybrid 3D-TOA/TDOA, PC-Intel_Core_i7-2.7GHz_RAM16GB 115
Figure 84. X, Y, Z Hybrid 3D-TOA/TDOA, Raspberry-Pi3ModelB1.4GHzRAM1GB.... 116
Figure 85. X, Y, Z Chan Alg, PC-Intel_Core_i3-1.7GHz_RAM4GB............................ 116

Figure 86. X, Y, Z Chan Alg, PC-Intel_Core_i7-2.7GHz_RAM8GB............................. 116
Figure 87. X, Y, Z Chan Alg, PC-Intel_Core_i7-2.7GHz_RAM16GB........................... 117
Figure 88. X, Y, Z Chan Alg, Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB 117
Figure 89. RSSI Multilateration method average execution time................................. 118
Figure 90. TOA Multilateration method average execution time.................................. 118
Figure 91. RSSI Direct method average execution time... 118
Figure 92. TOA Direct method average execution time.. 119
Figure 93. X, Y, Z error statistics TOA/TDOA Alg 119
Figure 94. Chan method average execution time... 119
Figure 95. Total Time Performance summary 121
Figure 96. Total Accuracy Error Performance summary 121
Figure 97. Random Walk output example 135

LIST OF TABLES

Table 1. Thesis Chapter Structure........ 3
Table 2. Positioning Technique Summary........ 20
Table 3. Localization Algorithms Summary........ 35
Table 4. Localization attack summary........ 42
Table 5. Secure Localization algorithms VS Localization attacks........ 53
Table 6. Localization frameworks summary. 54
Table 7. XVisor Memory Footprint........ 67
Table 8. Summary of the hypervisors that are presented in this paper. 88
Table 9. Localization algorithms simulated........ 92
Table 10. JSON input data simulation........ 105
Table 11. Nodes Coordinates input........ 108
Table 12. Performance Overview........ 117
Table 13. Execution Time Overview in milli seconds........ 120
Table 14. Algorithms Evaluation Summary........ 122

1. INTRODUCTION

The current research has as its goal to define how geo-location algorithms and schemes are utilized in order to achieve the location calculation of a node. Also, how to achieve such geo-location process as secure as possible from attacks. Finally, we present our proposed scheme for a secure and protected localization environment. In the current chapter an overview of the context and motivation behind the research will be given, as well as the adopted research method and approach. Also, an outline of the overall structure of the thesis will be given.

1.1 Motivation and objective

As a robust process, geo-location enables mobile app developers to identify the physical and real-world geographic location of various individuals and devices. A mobile developer can use geo-location to identify the exact latitudinal and longitudinal location of an internet-connected device through its GPS Location or Geo Tag. They can even us geo-location to make their mobile apps standout in the crowd and deliver richer user experience. There are also many reasons why most enterprises nowadays opt for geo-location-based mobile app development.

Geo-location allows users to share their physical location with your application if they choose to. Especially useful in social networking, geo tagging, and mapping, but applicable to any type of application, geo-location enables developers to enhance the user experience, making content, social graphs and advertisements more relevant to the location of the user. It is useful to track moving vehicles, such as planes or cars. In case of emergency, knowing were the location of trapped people in a burning building or during an earthquake can save a lot of lives. Geo-location can apply to many applications as we will see in the next chapters.

A system can adopt this geo-location concept to develop a safe environment were a user can access and utilize the system’s data only if he is inside a specific geographical scope. Imagine having a system that ensures that your personal data will be safe and accessed only by you and only in your house. We can have a system that has a specific geographical scope, where if a node enters this scope, the system can calculate its geological position and if it is authorized can access the system data. The system will be also responsible of the security and the data integrity of the user.

The main objective of this paper is to describe a proposed geo-localization framework, further information is provided at chapters 8-9-10, where it will be capable of:

1. Specify a geographical 3D field.
2. Calculate the position of a node when it enters the geographical fixed field.
3. Verify if the unknown node can access the framework data.
4. Give a safe access and interaction with the framework to the new nodes.
5. Ensuring that the data is full protected from attacks inside the fixed field.
6. Ensuring data integrity and no data leakage.

In order to achieve a framework with the above requirements, we must answer some specific questions where are presented in the next section.

1.2 Research questions

To reach the objective stated in the previous section, several questions need to be answered. The main research questions are as follows:

“How can we achieve geo-localization in a 3D space environment with low costs and high accuracy? “

As we will show in the next chapters there is a variety of algorithms and techniques that calculates the position of an unknow node in a system. However, there is a tradeoff between accuracy and energy consumption and hardware size. The more accuracy is needed the more energy and hardware is necessary in order to have high accuracies and low computations errors.

“Why to avoid the use of GPS in a geo-location system?”

GPS is expensive because it is a very slow communication channel. You need to communicate with three or four satellites for an extended duration at 50 bits per second. Using your GPS is a noticeable battery hog. New mechanisms and frameworks need to be adopted in order to achieve the position calculation of an unknown node with the minimal use of GPS in order to consume to the minimum of basic component of the modern devices. The battery.

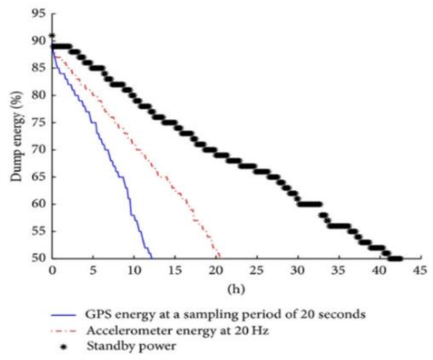


Figure 1. GPS Battery consumption.

“How a system can create a safe environment for a user that ensures data integrity? “

In order to answer this question, we adopted the concept of hypervisors as we will see on the next chapters. With the utilization of hypervisors, a system can create an environment (such as a Virtual Machine) where the user can be sure that no other can access its memory and data except from him.

“How can we ensure that the user’s data in the system will be fully protected? “

In most systems that data between nodes is transmitted there is a variety of attacks that can harm the user’s data and the system itself. We will see that we can utilize specific components for protection.

“Is there a system that calculate an unknow node position with low cost, providing a safe interaction between them and guarantees data protection and integrity, only inside a specific 3D geographical environment? “

As we will see on chapter 8, we propose a system where satisfies the requirement of the above question. A system that calculates unknow positions, provides safety and data integrity for users inside its scope.

1.3 Thesis structure and reading guide

The following table gives an overview of the research structure of this paper. We start with a short introduction in the geo-location process and application. We continue with a brief of positioning techniques and algorithms in order to achieve the position calculation process.

We present the main security problems in geo location systems alongside with some mechanisms and proposed schemes for protections. The hypervisors concept is then presented with documentation and implementation of some representative hypervisors.

Finally, our proposed framework is provided and our conclusions. In order to have a complete idea of our framework in chapters 6-7-8, make sure that you understand the basic concepts in chapters 2-3-4-5. Appendixes consists only of the implementation of some representative frameworks from chapter 5 and some source code presentation.

Table 1. Thesis Chapter Structure

Chapter	Subsections	Overview
1.Introduction	1.1 Context 1.2 Research setting 1.3 Objectives	An introduction of the concept and main objective of the paper and of its structure.
2.Introducing Geo-location	**2.1 Overview** 2.2 Why we need Geo-Location? 2.3 Geo-Location Applications	An introduction of the concept of geo-location and its applications. Why is so important? Why do we need to know where our position is in a system?
3.Introducing Positioning Techniques	3.1 Localization techniques Classification 3.2 Localization Algorithms	An introduction of the basic concept of localization techniques alongside with some localization schemes and framework.
4.Introducing Security in Geo-location	4.1 Security in Geo-location Systems 4.2 Attacks in Localization 4.3 Secure Geo-location Schemes	An introduction of the security and attacks in geo-localization applications alongside with some schemes and frameworks.
5.Hypervisors	5.1 Hypervisors Overview 5.2 XMHF- uberXMHF 5.3 Trust Hypervisor	A presentation of the Hypervisors framework and technology and some representative applications. How Hypervisors work, how the ensure security and data protection from attackers and the evaluation of those.
6. Proposed Secure Geo-Location Scheme	6.1 Introduction 6.2 Proposed Framework Overview 6.3 Proposed Framework Implementation	Presentation of a proposed framework for a secure localization environment.
7. Proposed Scheme Simulation	7.1 Framework Installation 7.2 Framework Execution	Simulation of the proposed framework.
8.Validation	8.1 Localization Algorithm Validation 8.2 Hypervisor Validation	Validation of the proposed framework.
9.Conclusions	Conclusion and Future work	Conclusions and future work.

1.4 Thesis concept

In order to present our proposed geo-localization framework, we need to understand first same basic concepts, technologies, mechanisms and algorithms that our framework is based on. Our proposed framework is a combination of two major concepts: Geo-Localization process and Security/Safety.

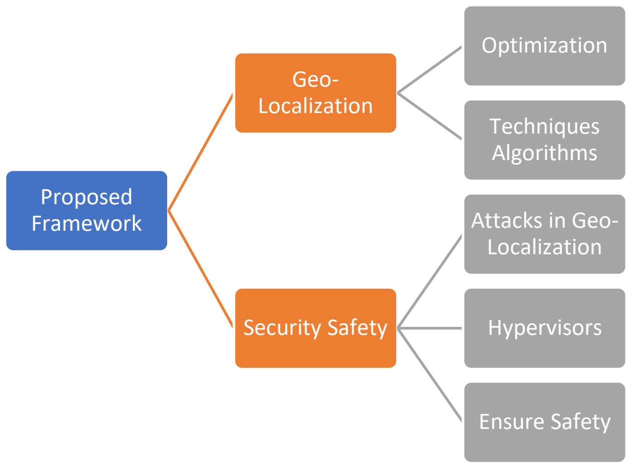
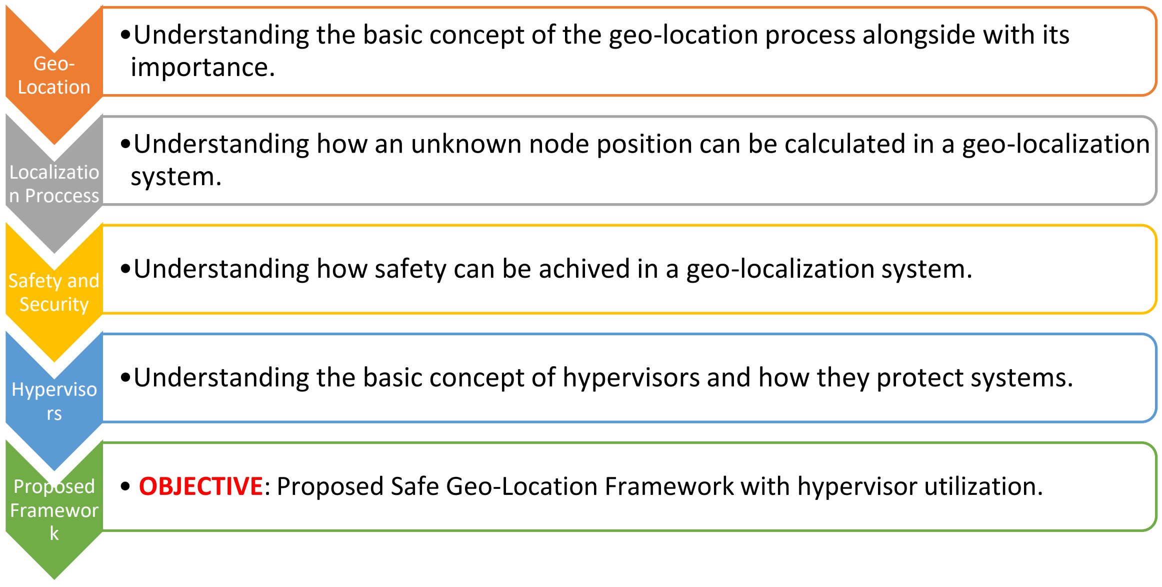


Figure 2. Proposed framework concept.

Firstly, is necessary to understand the basic concept of geo-location and how important is to modern applications alongside with how an unknown node position can be calculated in a geo-localization system (Chapter 2-3-4). Finally, is necessary to understand the concept of security and safety in such systems and how they can be protected from malicious attacks (Chapter 4-5).



- Understanding the basic concept of the geo-location process alongside with its importance.
- Understanding how an unknown node position can be calculated in a geo-localization system.
- Understanding how safety can be achived in a geo-localization system.
- Understanding the basic concept of hypervisors and how they protect systems.
- **OBJECTIVE**: Proposed Safe Geo-Location Framework with hypervisor utilization.

Figure 3. Thesis concept

2. INTRODUCING GEO-LOCATION

In this chapter we present a short overview on how geo-location is achieved in modern applications and technologies alongside with the basic concept of geo-location and localization.

2.1 Overview

In its simplest form geo-location, involves the generation of a set of geographic coordinates and is closely related to the use of positioning systems, but its usefulness is enhanced by the use of these coordinates to determine a meaningful location, such as a street address. The word geo-location also refers to the latitude and longitude coordinates of a specific location. The term and definition have been standardized by real-time locating system standard ISO/IEC 19762-5:2008 [1]

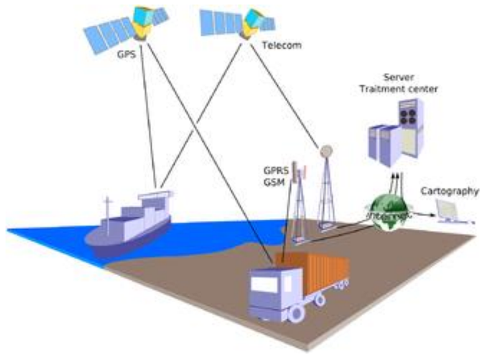


Figure 4. GPS Geo-location

For either geolocating or positioning, the locating engine often uses radio frequency (RF) location methods, for example Time Difference Of Arrival (TDOA) for precision. TDOA systems often use mapping displays or other geographic information system. More details for those techniques are presented in chapter 3. When satellite navigation (such as GPS) signals are unavailable, geo-location applications can use information from cell towers to triangulate the approximate position, a method that is not as accurate as GPS but has greatly improved in recent years. This is in contrast to earlier radiolocation technologies, for example Direction Finding where a line of bearing to a transmitter is achieved as part of the process.

Internet and computer geo-location can be performed by associating a geographic location with the Internet Protocol (IP) address, MAC address, RFID, hardware embedded article/production number, embedded software number (such as UUID, Exif/IPTC/XMP or modern steganography), invoice, Wi-Fi positioning system, device fingerprint, canvas fingerprinting or device GPS coordinates, or other, perhaps self-disclosed information. IP address location data can include information such as country, region, city, postal/zip code,[6] latitude, longitude and time zone.[7] Deeper data sets can determine other parameters such as domain name, connection speed, ISP, language, proxies, company name, US DMA/MSA, NAICS codes, and home/business. At times geo-location can be more deductive, as with crowdsourcing efforts to determine the position of videos of training camps, combats, and beheadings in Syria by comparing features detected in the video with publicly available map databases such as Google Earth, as practiced by sites such as Bellingcat Some standards and name servers include:

ISO 166, FIPS, INSEE, Geonames, IATA and ICAO. For geographic locations in the United States, the American National Standards Institute (ANSI) Codes are often used. [11] ANSI INCITS 446-2008 is entitled "Identifying Attributes for Named Physical and Cultural Geographic Features (Except Roads and Highways) of the United States, Its Territories, Outlying Areas, and Freely Associated Areas, and the Waters of the Same to the Limit of the Twelve-Mile Statutory Zone".[11] A number of commercial solutions have been proposed:

- WOEID (Where on Earth IDentifier) is a unique 32-bit reference identifier that identifies any feature on Earth. [2]
- NAC Locator provides a universal geocoding address for locations on the planet. [3]

2.2 Why we need Geo-location?

Identifying our exact location on earth has been a fascination of mankind since the Ancient Greeks used the stars to triangulate their position. The importance of knowing the exact position of o person, house, car, city, hospital etc. in the whole world with precision is obvious and today, we have a wide-array of location-based services that have made their way into office buildings and jean pockets everywhere. It is important because the Geo-location systems help an average person to locate herself precisely anywhere on the planet without having to be too much technically literate and for free. In fact, it has been rightly identified as the backbone for several businesses, without which it will be nearly impossible for the owners to operate it.

Knowing the exact position location can be used for a number of applications and also can be useful in many emergency situations. It allows you to locate yourself on the high seas or featureless Saharan desert where there may not be any landmarks to orient yourself because it does not depend on any terrestrial system (such as its predecessor LORAN which is space based). It also can assist for rescue of fellow humans in situations such as natural disasters like earthquakes in a timely manner which was not possible before.

Localization systems can encourage the innate desire of humans to explore the unknown lands without the fear of not being able to return since their global position is calculated. It can quietly help to protect soldiers in times of conflict and in hostile lands by helping them navigate themselves and by helping others to find them if needed.

Automotive industries utilize geo-location application to provide rout guidance services to drivers. Internet and cloud application need the exact position of their user in order to provide their services with high speed and performance. Geo-location data can be utilized for statistics and demographics researches and applications. Space and aviation industries use geo-location data for calculate routes. The need of geo-location systems is continuously growing and the majority of applications and industries require the exact location of their users in order to provide their services with accuracy, high performance and safety.

Geolocation commonly uses Global Positioning System (GPS) and other related technologies to assess and specify geographical locations. The global positioning systems (GPS) market size was estimated at USD 37.9 billion in 2017. It is anticipated to progress at a CAGR of 18.4% during the forecast period.

Increasing penetration of smart phones along with rising GPS-enabled vehicles is projected to bolster the growth of the market during the forecast period. Moreover, surging use of social media across developing countries and a high number of mergers and acquisitions between component manufacturers and integrators are poised to stoke the growth of the global positioning systems market.

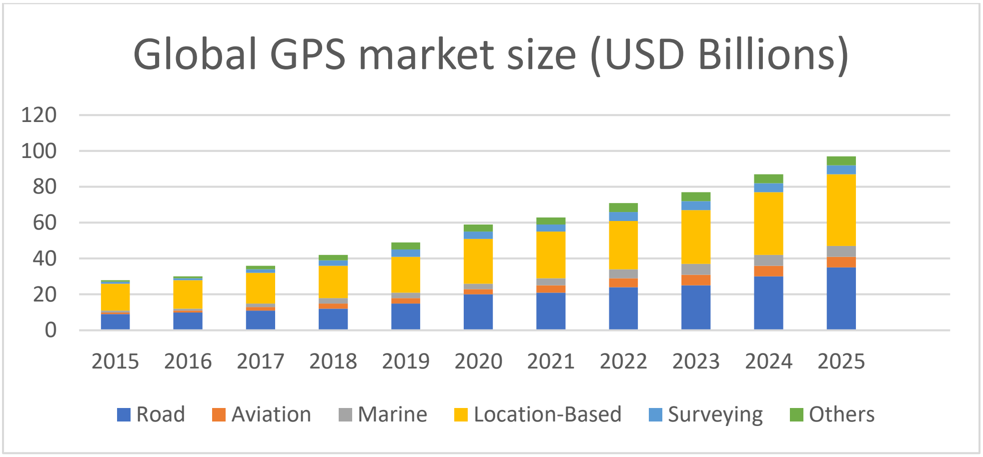


Figure 5. GPS Market Size, from Grand View Research

2.1.1 The History of Geo-location

With all of the progress made and all of the efficiencies introduced, geo-location's history is uniquely fascinating. Let's take a look at how far we've come:

- **2,000+ Years Ago** – Ancient Greeks triangulated their geographical location using only the stars.
- **1933** – Radar is finally on the map, as the U.S Naval Research Laboratory's Leo Young, proposed the use of a pulse radar technique to be able to detect aircraft and ships.
- **1940** – The Naval Research Laboratory enabled the first submarine to use radar. It had a 20-mile range.
- **1957 –** *Sputnik I,* the first artificial satellite to go into space, is launched by the Soviet Union. It was about the size of a beach ball at 58 cm., weighed only 184 pounds, and took about 98 minutes to orbit the Earth on its elliptical path.
- **1973 –** The *Navstar Global Positioning Satellite (GPS)* system is proposed by the Pentagon.
- **1978 –** The first 11 *Navstar GPS* satellites are launched into space.

- **1983** – President Ronald Reagan offers to let all civilian commercial aircraft use the GPS system to improve air navigation safety. Although, the commercial use of GPS is granted with “selective availability,” or restricted use.
- **1989** – *Magellan* becomes the first to sell a hand-held navigation device with the release of the *Magellan NAV 1000.*
- **1993** – The U.S. Air Force sends the final *Navstar* GPS satellite into orbit, completing an entire network of 24 Global Positioning System (GPS) satellites. Although, they were not yet fully operational.
- **1995** – The *Navstar GPS* program reached a fully operational status in April of 1995. The project cost 10 to 12 billion dollars.
- **1999** – The first commercial use of GPS – a safety phone called the *Benefon Esc!* – is released for consumer purchase. It is sold mainly in Europe.
- **2000** – President Bill Clinton lifts “selective availability” on GPS-usage for civilians. This allows civilians and consumer-facing services to use GPS with the same pinpoint accuracy as the military.
- **2002** – *Nikon* introduces their *D1H & D1X* camera models – the first DSLR cameras to offer a GPS interface.
- **2004** – Up to 15 percent of U.S. farmers use GPS-controlled tractors and/or combines and are saving as much as 5 percent in fertilizers and pesticides with the use of precision guidance systems.
- **2005** – *GIOVE-A*, Europe’s first experimental GPS satellite is launched into space, serving to test critical technologies for the *Galileo* program, meant to give Europe independence from the U.S, Russian and Chinese GPS systems. The program is scheduled to be fully operational in 2020.
- **2005** – *Google Maps* officially debuts!
- **2005** – *Yelp* popularizes location-based reviews leveraging geo-location technologies.
- **2006** – After testing equipment, such as the handheld *SkyCaddie,* and being courted by the electronic industry for several years, the *U.S. Golf Association* permits distance-measuring GPS devices and laser range finders.
- **2006** – *Geopointe* is founded, revolutionizing the way *Salesforce* users can visually leverage their data.
- **2007 –** GPS becomes a multi-billion-dollar industry, causing further restrictions to be lifted on commercial usage by President George W. Bush.
- **2009** – *Foursquare* launches, helping further popularize geo-location review services.
- **2012** – The U.S Supreme Court rules that warrantless GPS tracking is constitutional.
- **2015** – *Facebook* begins licensing location-based data from Factual, a geodata platform. This includes “US Places data,” which includes hotels and restaurants “extended attributes,” to support *Facebook business page*s, check-ins, Places search and more.
- **2017** – *Match.com* reportedly enables users to see which other users they have crossed real-life paths with.
- **2017** – *Facebook* fights to represent the primary “digital presence” for local businesses. With 2.5 billion comments being added to *Facebook business Pages* every month, they are now competing directly with *Google*.

2.1.2 Applications

2.1.2.a Aviation

Most of the modern aircraft use geo-location receivers to provide the pilots and passenger with real-time aircraft position. They also provide a map of various destinations depending on where the aircraft operates. This application is also used by the airline operators to decide which route is the fastest, safest, and most fuel-efficient among the destinations. They also use the app to track the aircraft and direct the pilot in the case where there is a change in the weather conditions or any other issue that may arise.

2.1.2.b Marine

Highly accurate navigation app is needed by boat captains to enable them to navigate through waters to their destinations. These applications ensure that the channels are clear and there are no obstacles that can hinder their navigation. They are also required in the marine departments since they are used to map and position dredging operations in rivers, sandbars, and wharves to ensure that other boats are aware of how deep they should get.

2.1.2.c Farming

Farmers have a specific season for planting, weeding, and harvesting, and due to the repeat in the seasons, they put the geo-location receiver on their tractors and other farming equipment. This allows them to map their plantations and ensure that they return to precisely the same time when planting or weeding in the next season. This strategy is effective especially in seasons when it’s foggy with less visibility since the machine will still operate since its geo-location and not visual reference guides it. More so, its high accuracy makes it suitable for use in mapping soil sample locations, and the farmers can locate the areas that have soils suitable for farming.

2.1.2.d. Science

This is one field that intensively uses the geo-location app especially since there are numerous departments in the science field, this include physics, biology and earth science to mention a few. Before the invention of this app, scientists used metal and plastic bands to tag animals, and they would follow them to various location while monitoring them. However, since the invention and enhancement of the navigation app, it has helped scientist to fit the animal with the collars and the app can automatically record the animal’s movement and the information is transmitted to a through a satellite to the researcher. This means that they can trace the location of the animals’ movement without having to relocate them physically. Most of the sciences taught in schools will have more practical evidence especially since geo-location provides accurate data. Earth scientist also uses the app to study how landscapes change over time. Or any geographic area that they may be interested.

2.1.2.e Surveying

Surveying is one of the uses of geo-location that are essential especially since it is used in mapping and measuring various measures on the earth surface and underwaters. It is used in determining land boundaries, mapping sea floors, and highlighting the changes in the shape of structures. The best thing about high accuracy geo-location app is that the surveyors can set it up over a single point and establish a reference marker. They can also use it in a moving configuration to map the boundaries of particular features. With the data obtained from the application, they can easily key in the details into a software that will help them offer their customers with a detailed chart.

2.1.2.f Military

The US Department of Defense was the first to develop the geo-location GPS app system, and since then the system has been adopted by numerous military forces around the world. Other countries have even decided to develop their satellite navigation networks as a defense mechanism during war times. Today, there has been a diverse use of the app, and it can be used to map the location of vehicles and other machinery such as missiles during a war. This is a technique used purposely to protect the soldiers and also manage resources.

2.1.2.g Financial Services

Financial organizations such as banks use this app to schedule and determine local and international money transfers. They are also using it to provide audit trails of financial transactions. More so, since more than 80% of the transactions are made through debit and credit cards, it has been easier to provide a higher level of timing accuracy. The geo-location GPS satellite is necessary for the financial field since it allows for data and time stamps of Electronic Funds Transfers.

2.1.2.h Telecommunications

Telecommunications especially the mobile telephones use this app to provide its users with accuracy, reliability, and stability of their operations. Although other clocks can provide this, the geo-location supports the derivation of synchronized time zones with the help of the satellite signals.

2.1.2.i Heavy Vehicle Guidance

Heavy tack machines used in mining and constructions also use this technology. For example, in highway construction, the marker pegs and surveyors have been replaced by the improvised in-cabin vehicle guidance and control systems. This makes the work easier since the driver only needs to follow the surveyor's pre-programmed site plan.

2.1.2.j Road Transportation

Majority of users of this technology are taxi services, emergency vehicle location, commercial fleet management and freight tracking, public transport monitoring, dispatch, and navigation. Private car owners also use the app, and most of the new car models come with a factory-fitted GPS.

2.1.2.m Social Activities

Some of the social activities that have incorporated the use of this technology include cross-country cycling, skiing, hiking skydiving, paragliding, geotagging photographs, geocaching, geo dashing among others.

2.1.2.n Easy Access to Emergency Roadside Support

In case of an accident or an emergency, you can seek assistance using the pre-programmed emergency numbers on your smartphone. The best thing about using this app is that the emergency crew can trace your current location without having to provide any details.

2.1.2.l Public Safety and Disaster Relief

The best thing about geo-location is that it can be used in any weather or environmental condition which is the reason why is preferred for use during disaster management. The emergency vehicles and supplies are tracked using geo-location.

2.1.2.o Can be used by Disabled People

People with special needs are at times left on their one when their caretakers and loved ones have to work. The geo-location tracker is not only used to track their location, but it can be of great importance in the case of emergencies.

2.1.2.p Civil engineering applications

Civil engineering works are often done in a complex and unfriendly environment, making it difficult for personnel to operate efficiently. The ability of geo-location to provide real-time submeter- and centimeter-level accuracy in a cost-effective manner has significantly changed the civil engineering industry. Construction firms are using geo-location in many applications such as road construction,

2.1.2.q Space and Spacecraft

Some near and far term space missions involve formation flying, which requires that the positions of multiple spacecraft be accurately known relative to a hub spacecraft.

2.1.2.r Security

The location base encryption or Geo-Encryption technique uses GPS technology to enhance the data security. This concept is developed so that at a particular position and time, the specific recipient will decrypt the files. The breaching of data due to stolen laptops are a major problem. This technology will be used to restrict unauthorized user for any violation. Also, it can assure that data can be secure in specific geological fields and can be accessed only in the user is inside those specific fields. For instance, files and data from a specific university can be accessed only inside the university campus.

3. INTRODUCING LOCALIZATION TECHNIQUES

In this chapter we present a short overview on localization technologies and techniques and how geo-location is achieved in those.

3.1 Localization techniques classification

Many of these applications need location base services. Although GPS is a direct solution to the localization problem, the high cost, high power consumption, and poor performance of GPS inside an indoor environment have necessitated the research on localization algorithms. Over the past few years, the scientific world has observed a lot of research efforts on this topic. Note that the localization is defined as the determination of the position of an unknown node, sometimes with the help of nodes with known position, and at other times using the connectivity information between the unknown nodes.

Recent studies have investigated the effect of mobility in localization, real world applications, “Anchor Based” and “Anchor Free” localization methods , “Range Based” localization algorithm (distance measurement technique to calculate the location of unknown nodes) and “Range Free” localization algorithm (connectivity rather than distance) , “Cooperative” (communication exists among all nodes) and “Non-Cooperative” (unknown nodes communicate only with the anchor nodes) algorithms, “Centralized” algorithm based localization (*aka* network-centric positioning) and “Distributed” algorithm (no central control on the determination of the node’s position and each node estimates its location based on the locally gathered information - *aka* “self-positioning” algorithm

3.1.1 GPS Based Localization

Global Positioning System (GPS) localization has been attracting attention recently in various areas, including intelligent transportation systems (ITSs), navigation systems, road tolling, smart parking, and collision avoidance. Although, various approaches for improving localization accuracy have been reported in the literature, there is still a need for more efficient and more effective measures that can ascribe some level of accuracy to the localization process.

The Navigation Satellite Time and Ranging (NAVSTAR) Global Positioning System (GPS) is a worldwide radio-navigation system created by the U. S. Department of Defense (DOD) to provide navigation, location, and timing information for military operations. System testing using a limited number of satellites began in 1978 with the system being declared fully operational in 1995. The system was declared available for civilian uses in the 1980s and has seen burgeoning civilian application for navigation and mapping. GPS is the U.S. implementation of a Global Navigation Satellite System (GNSS). Increasingly, GPS receivers have the capability to utilize signals from other GNSS such as the Russian GLONASS or European Galileo systems. SESD has no limitations on the use of signals from other GNSS.

The accuracy of the basic GPS system is approximately 15m. GPS accuracy can be affected by a number of factors including the Selective Availability feature, atmospheric delays, satellite clock and orbit errors, multipath signals, signal strength, and satellite geometry relative to the user. Wherever the node is on the planet, at least four GPS satellites are 'visible' at any time. Each one transmits information about its position and the current time at regular intervals. These signals, travelling at the speed of light, are intercepted by your GPS receiver, which calculates how far away each satellite is based on how long it took for the messages to arrive. Once it has information on how far away at least three satellites are, the GPS receiver can pinpoint the location using a process called trilateration.

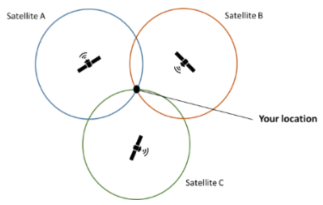


Figure 6. GPS Trilateration Model

3.1.2 GPS Free Localization

With a network of thousands of nodes, it is unlikely that the position of each node can be precisely predetermined [4]. Although GPS based localization schemes can be used to determine node locations within a few meters, the cost of GPS devices and the non-availability of GPS signals in confined environments prevent their use in large scale sensor networks.Below is presented the recent advances on localization techniques in WSNs by considering a wide variety of factors and categorizing them in terms of data processing (centralized vs. distributed), transmission range (range free vs. range based), mobility (static vs. mobile), operating environments (indoor vs. outdoor), node density (sparse vs. dense), routing, algorithms, etc.

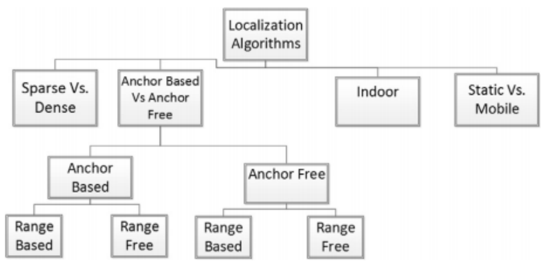


Figure 7 .Classification of localization techniques

3.1.2.a Anchor Based/Centralized

In Centralized algorithms all computation is done in central server. Centralized algorithms resolve computational limitations of nodes. In these algorithm nodes have to communicate to BS, unfortunately communication consumes more energy than computation. In distributed algorithms computation is distributed among sensor nodes.

In these algorithms only Inter node communication is done which consumes less energy as compared to communication cost in centralized algorithm.

3.1.2.b Anchor Free/Decentralized

The main advantage of a decentralized data fusion system is the lack of dependency of the whole system on a central processing unit. Also, in such systems, the malfunctioning of a sensor will not affect the whole estimation, and local estimators with functioning sensors continue to perform properly. Several general decentralized data fusion algorithms are presented in [6-8]. In these works, a network of sensors with several local estimators is considered and several local estimates are produced.

In [7], a decentralized architecture with applications in the navigation of vehicles is presented. The method is also applicable to the cases where the sensor measurements are asynchronous. In [8], the localization in a known environment is addressed and simulation results are provided to show the estimation results in a decentralized architecture. Recently, developing decentralized algorithms for SLAM with special application on the navigation of a team of robots has attracted new attentions [9]. In these works, the estimated landmark positions are transmitted between the robots for updating the map of the environment.

3.1.2.c Range Based

Range-based estimate location by point-to-point distance measurements [10]. Some common distance measurement methods are angle of arrival (AoA), time of arrival (ToA), time difference of arrival (TDoA), acoustic energy, and received signal strength indicator (RSSI). The first three methods require complex hardware set up while RSSI is simpler than the others but less accurate. After gathering the information of anchors and sometimes of other unknown nodes, distances are combined using techniques like trilateration or particle filter etc.

3.1.2.d Range Free

Range free localization algorithms use connectivity information among the nodes to determine the positions of unknown nodes [11]. Since the range base methods require a hardware setup that is both complex and costly, a range free method can be a possible solution to hardware limitation problems.

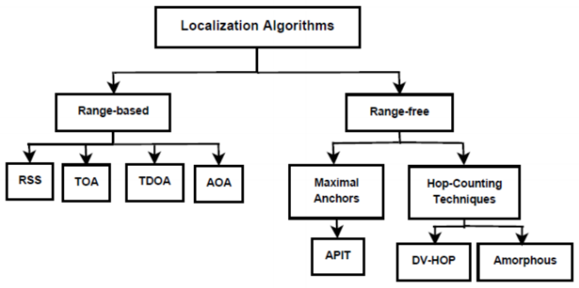


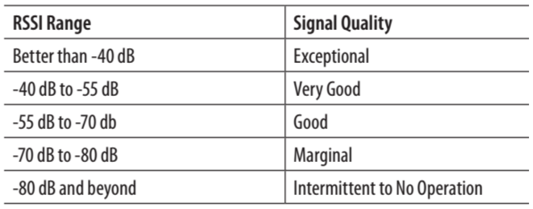
Figure 8. Range Based Vs Range Free

3.1.3 Range Base Positioning techniques

3.1.3.a RSSI

The Received Signal Strength Indicator (RSSI) value is part of the data packet transmitted by all Veris Aerospond sensor units [12]. It is intended as means to obtain a relative indication of the quality of connection that exists between the sensor unit and the access point it is connected to on the wireless network. In order for this to be a useful tool in determining the quality of the connection, there are a few principals that need to be understood about what the RSSI value means. The RSSI formula is presented below:

Signal Strength Signal strength is based on a number of factors, including the output power of the transmitter (the original strength of the signal), the sensitivity of the receiver (how well the receiving device can hear weak signals), the gain of the antennae at both ends of the path, and the path loss, or attenuation of the signal as it travels through the air from the transmitter to the receiver. Signal strength is expressed in units of decibels (dB). Due to the low power levels and the attenuation of free space, an RSSI value is expressed as a negative number. The more negative the number, the weaker the signal strength; conversely the closer the number is to zero, the stronger the signal.



RSSI Range	Signal Quality
Better than -40 dB	Exceptional
-40 dB to -55 dB	Very Good
-55 dB to -70 db	Good
-70 dB to -80 dB	Marginal
-80 dB and beyond	Intermittent to No Operation

Figure 9. RSSI range performance

Given the received signal power PR (in Watt), the transmitted signal power PT (in Watt), the receiver's antenna gain GR, the transmitter's antenna gain GT , the signal wavelength λ, the distance d in meters and the signal propagation constant n, the Friis' equation is defined as

$$P_R = P_T \frac{G_T * G_R * \lambda^2}{(4 * \pi)^2 * d^n} \quad (1)$$

Since the RSSI value is expressed in dBm, we have to convert the results of Equation 1 from Watts to dBm using:

$$\mathrm{P[dBm]} = 10 \cdot \log_{10} (\mathrm{P[W]} \cdot 10^3) \ (2)$$

and with Equation 2 we can obtain a relation between distance and receive power, simplified for the case of a 1-meter reference distance as:

$$\mathrm{RSSI} = \mathrm{A} - 10 \cdot \mathrm{n} \cdot \log_{10} \mathrm{d} \ (3)$$

where A is the received power in dBm with the two antennas 1 m distant, and n the loss parameter (or loss exponent) of the specific environment. Hence, the distance d can be easily calculated as

$$d = 10^{\frac{A-RSSI}{10*n}}$$

3.1.3.b TOA

Time of Arrival [13] is the simplest and most common ranging technique, most notable used in the Global Positioning System (GPS). This method is based on knowing the exact time that a signal was sent from the target, the exact time the signal arrives at a reference point, and the speed at which the signal travels. Once these are known, the distance from the reference point can be calculated using the simple equation.

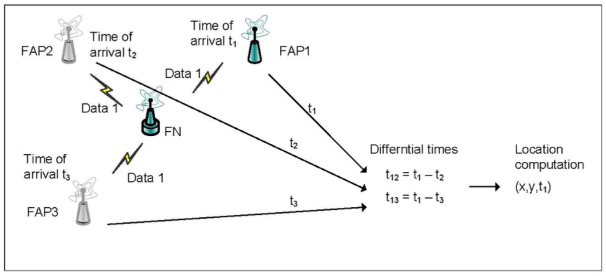


Figure 10. TOA Overview

Using this distance, the set of possible locations of the target can be determined. In two dimensions, this yields a circle with the equation:

$$d = c * (t_{arrival} - t_{sent})$$

$$d = \sqrt{(x_{ref} - x)^2 + (y_{ref} - y)^2}$$

where c is the speed of light and (x_{ref}, y_{ref}) is the known position of the reference point. Once this set is calculated for enough reference points (at least three for two dimensional or at least four for three-dimensional), the exact position of the target can be calculated by finding the intersection.

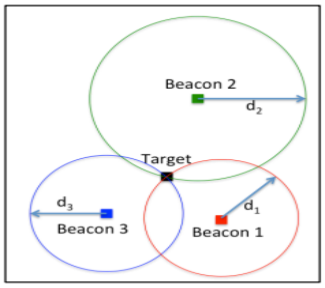


Figure 11. TOA Trilateration

In this example, the Target (black) is surrounded by three Beacons (red, green, and blue). At time t1, a signal is sent from Beacon 1 to the Target, which is received at t2. The distance (d1) between the Target and Beacon 1 is calculated, then the circle of possible locations is drawn, a. This process is repeated for Beacons 2 and 3, which yields two more circles, as shown in Figure 3.

3.1.3.c TDOA

Time Difference of Arrival is the second-most popular ranging technique, and it is somewhat more versatile than ToA [14]. This method does not require the time that the signal was sent from the target, only the time the signal was received and the speed that the signal travels. Once the signal is received at two reference points, the difference in arrival time can be used to calculate the difference in distances between the target and the two reference points. This difference can be calculated using the equation:

$$\Delta \mathrm{d} = \mathrm{c} * (\Delta \mathrm{t})$$

where c is the speed of light and Δt is the difference in arrival times at each reference point. In two dimensions, this leads to the following equation [2]:

$$\Delta \mathrm{d} = \sqrt{(\mathrm{x}_2 - \mathrm{x})^2 - (\mathrm{y}_2 - \mathrm{y})^2} - \sqrt{(\mathrm{x}_1 - \mathrm{x})^2 - (\mathrm{y}_1 - \mathrm{y})^2}$$

where (x_1, y_1) and (x_2, y_2) are the known positions of the beacons. Using nonlinear regression, this equation can be converted to the form of a hyperbola [2]. Once enough hyperbolas have been calculated, the position of the target can be calculated by finding the intersection.

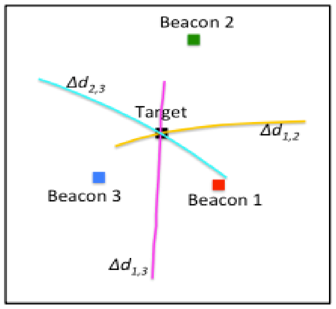


Figure 12. TDOA Trilateration

In this example, we have the same setup of a target (black) surrounded by three beacons (red, green, and blue). A signal is sent from the Target at an unknown time, which is received by Beacon 2 at t1 and Beacon 3 at t2. The difference in distance (Δd) is calculated, and the hyperbola of possible locations is drawn, as shown in Figure 5.

This hyperbola will have two branches, which would normally make finding the intersection more difficult. However, if the approximate location of the target is known (e.g. through a previously measured location), one of the branches can be discarded. In this case, the top branch is discarded. This process is repeated with the remaining Beacon pairs, and the result is shown in Figure 6.

3.1.3.d AOA

AOA is defined as the angle between the propagation direction of an incident wave and some reference direction, which is known as orientation [14]. *Orientation*, defined as a fixed direction against which the AOAs are measured, is represented in degrees in a clockwise direction from the North. When the orientation is 0◦ or pointing to the North, the AOA is absolute, otherwise, relative. One common approach to obtain AOA measurements is to use an antenna array on each sensor node.

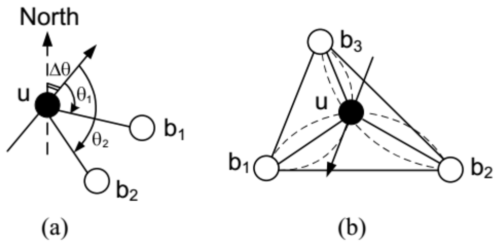


Figure 13. AOA Trilateration

3.1.4 Range Free Positioning techniques

3.1.4.a DV hop

The Distance Vector-Hop (DV-Hop) algorithm was proposed as a distributed localization algorithm based on distance vector between nodes [15]. Rather than estimating the distance between neighboring wireless nodes by using ranging methods, this algorithm first calculates the actual distance between every pair of anchor nodes (since the coordinates of these nodes are known a priori). It then finds hop-distance (i.e., the number of hops) between every pair of anchors and calculates the average distance per hop along these paths. These distances are then used for localization of sensor nodes. Specifically, the localization process while using DV-Hop algorithm consists of the following three phases:

Step 1: Calculating the minimum hop count between beacon nodes and unknown nodes. Beacon nodes broadcast information which shows their positions to neighbouring nodes by using the classical distance vector routing protocol. The information contains $\{id, x_i, y_i, H_i\}$, where id, (x_i, y_i), and H_i represent the identifier, the coordinate, and the hop count of beacon nodes , respectively. Moreover, the initial value of H_i is set to zero.

The nodes receiving the broadcast information record the localization and hop counts of beacon nodes as vectors, which are then transmitted to neighbouring nodes (the value of hop count is incremented by one). When a node receives the same id group, it is supposed to compare the newly obtained value of H_i with the original value and then select the minimum value to replace and update the original group; otherwise, the newly obtained group is abandoned. The position information and minimum hop count of all beacon nodes are obtained by this communication mode in WSNs.

Step 2. Estimating the average hop distance. The purpose of calculating the average hop distance and minimum hop count first is to estimate the distance between unknown nodes and beacon nodes. After acquiring the localization and the hop count of beacon nodes in the first stage, the average hop distance of whole networks can be computed.

The information is then broadcast to the whole network, or all networks. Furthermore, most nodes are required to receive the average hop distance from their nearest beacon nodes. The distances between beacon nodes and unknown nodes can be calculated by multiplying the average hop distance by the hop count. Here, hd_i and h(ij) denote the average hop distance and the hop distance between a be1acon node and an unknown node H_i respectively, as shown in the following formula:

$$hd_i = \frac{\sum \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum h(ij)}$$

The distances between unknown nodes and beacon nodes are calculated using the following formula:

$$d_i = hd_i \text{ x Hop,}$$

where hd_i signifies the average hop distance, while Hop is the minimum hop count between unknown nodes i and beacon nodes.

Step 3. Based on plane geometry, the coordinates of unknown nodes can be acquired in the case of knowing the coordinates and distances between three beacon nodes. Suppose that the coordinates of three beacon nodes are (x_1,y_1), (x_2,y_2) and (x_3,y_3) respectively, and the distances between these three beacon nodes and an unknown node D are expressed as d_1, d_2, and d_3 separately, then, the following formula is obtained:

$$\begin{cases} (x_1 - x)^2 + (y_1 - y)^2 = {d_1}^2 \\ (x_2 - x)^2 + (y_2 - y)^2 = {d_2}^2 \\ (x_3 - x)^2 + (y_3 - y)^2 = {d_3}^2 \end{cases}$$

Meanwhile, the coordinate of node D can be calculated by using the following formula:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2(x_1 - x_3) & 2(y_1 - y_3) \\ 2(x_2 - x_3) & 2(y_2 - y_3) \end{bmatrix}^{-1} \begin{bmatrix} {x_1}^2 - {x_3}^2 + {y_1}^2 - {y_3}^2 + {d_3}^2 - {d_1}^2 \\ {x_2}^2 - {x_3}^2 + {y_2}^2 - {y_3}^2 + {d_3}^2 - {d_2}^2 \end{bmatrix}$$

In this way, the coordinates of unknown nodes can be computed.

3.1.4.b APIT

The approximate point in triangle (APIT) is an approach for area estimation [16]. APIT algorithm requires a small percentage of anchors and employs a novel area-based approach to perform location estimation by segmentation of the field. Moreover, these nodes can be equipped with high-powered radio transmitter. The main idea of APIT for localization of nodes is to consider overlapping triangles. The vertices of these triangles are anchors.

Bounding triangles are obtained using any group of three reference nodes, rather than the coverage area of a single node. In the APIT algorithm, the sensor nodes receive location information from the nearby anchors initially.

Second, the point in triangulation (PIT) test checks whether a sensor node is in a virtual triangle that is formed by connecting the three anchors from which signals are received. After the PIT test is done, the APIT algorithm aggregates the results through a grid SCAN algorithm [45]. The APIT algorithm calculates the CoG (Centre of gravity) of the intersections of all the overlapped triangles in which the node resides to determine its location

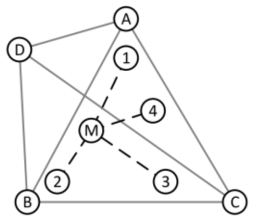


Figure 14. The APIT principle

In Fig. 9, M is the unknown node, the point A, B, C and D are four anchor nodes which are received by M, and point 1, 2, 3 and 4 are the neighbor nodes which received by M. The point A, B, C and D form four triangles, namely △ABC, △ACD, △ABD, and △DBC.

In △ABD, the unknown node M has the neighbor node 3 is away from the vertices A, B and D of the triangle at the same time compared with the unknown node M. Therefore, according to the principle of APIT, the unknown node M is judged to be outside of △ABD. Similarly, the unknown node M is located inside of △ABC, △DBC and outside of △ACD

Table 2. Positioning Technique Summary

Technique	Cost	Accuracy	Energy efficient	Hardware size
GPS	High	High	Less	Large
GPS Free	Low	Medium	Medium	Small
Centralized	Depends	High	Less	Depends
Decentralized	Depends	Low	High	Depends
RSSI	Low	Medium	High	Small
TOA	High	Medium	Less	Large
TDOA	Low	High	High	Less Complex May be Large
AOA	High	Low	Medium	Large
DV hop	Low	Medium	High	Small
APIT	Medium	Medium	High	Medium

3.2 Localization Algorithms

3.2.1 FP-MPP-APIT

The proposed algorithm is an approximate point in triangulation test (APIT) -based localization algorithm combining the Fermat point and the mid-perpendicular plane models, and it is named FP-MPP-APIT [17]. It is a distributed range-free localization algorithm for three-dimensional WSNs that possesses the advantages of APIT-based localization algorithms. In geometry, the Fermat point of a triangle, also called the Torricelli point or Fermat–Torricelli point, is a point such that the total distance from the three vertices of the triangle to the point is the minimum possible. The Fermat point of a triangle with largest angle at most 120° is simply its first isogonic center or X(13), which is constructed as follows: Construct an equilateral triangle on each of two arbitrarily chosen sides of the given triangle. Draw a line from each new vertex to the opposite vertex of the original triangle. The two lines intersect at the Fermat point.

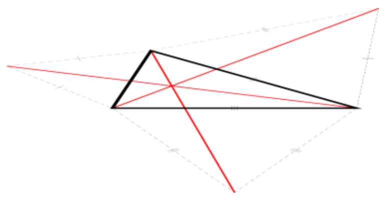


Figure 15. Fermat Point in Triangle

This point can divide a triangular pyramid into four sub-triangular pyramids. Thus, the APIT-3D algorithm can be used to determine in which sub-triangular pyramid the unknown node *M* is located. The sub-triangular pyramid that contains the unknown node is called the available sub-triangular pyramid. The mid-perpendicular plane model is used to divide the available sub-triangular pyramid.

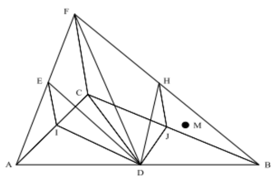


Figure 16. Mid-Perpendicular plane model

The steps of this algorithm are described below:

a. All beacon nodes broadcast their positions and ID.
b. Each unknown node maintains a counter k and in every received beacon message the counter adds one and records the message.
c. If the unknown node has more than four constructs a triangular pyramid APIT-3D algorithm is used to determinate if the node is located in the pyramid.
d. The Fermat point model is adopted to divide the pyramid in four sub-pyramids.

e. The mid-perpendicular plane model is used to divide the available sub-triangular pyramid into a set of irregularly shaped subplaces.
f. The unknown node is located.

3.2.2 3D-IDCP

Is an improved proposed 3D localization algorithm and a 3D localization model to improve positioning accuracy and coverage [18]. The proposal utilizes the range base localization and includes four phases. The concept of Degree of Coplanarity is added and the best positioning unit based on the Degree of Coplanarity is selected to ensure positioning accuracy. In addition, the unknown nodes which have been located are promoted to assistant anchor nodes. Degree of Coplanarity (DCP) represents the coplanar degree of four anchor nodes in the three-dimensional space. Degree of Coplanarity (DCP) can be expressed with

$$\mathrm{DCP} = \begin{cases} 0 \\ \rho \end{cases}$$

where 0 means that the space is coplanar and ρ else, DCP ∈ (0, 1].

In the three-dimensional plane a positioning unit consists of at least 4 anchor nodes. DCP is utilized to represent the coplanar degree of the four nodes. The radius ratio of a tetrahedron us utilized ρ. The DCP ensures that the positioning unit can be selected. The position of the unknown node can be calculated through quadrilateration. 3D-IDCP is utilized to promote unknown nodes that are located to assistant nodes. The steps of this algorithm are described below:

a. Choose the best positioning unit based on the DCP value to locate the unknown nodes.
b. Promote the unknown node that have been located to beacon nodes and broadcast their positions.
c. Locate now unknown nodes.

The Quasi-Newton method is used for nonlinear optimization.

3.2.3 Novel Centroid

Bulusu and Heidemann have proposed the centroid localization algorithm, which is a range-free, proximity-based, coarse-grained localization algorithm [19] [20]. The algorithm implementation contains three core steps. First, all anchors send their positions to all sensor nodes within their transmission range. Each unknown node listens for a fixed time period *t* and collects all the beacon signals it receives from various reference points. Second, all unknown sensor nodes calculate their own positions by a centroid determination from all *n* positions of the anchors in range. The centroid localization algorithm, which uses anchor nodes (reference nodes), containing location information (*xi* ,*yi*) , to estimate node position. After receiving these beacons, a node estimates its location using the following centroid formula:

$$(x_{est}, y_{est}) = \left(\frac{x_1 + \cdots + x_N}{N}, \frac{y_1 + \cdots + y_N}{N}\right)$$

The steps of this algorithm are described below:

a. All beacon nodes broadcast their positions. Each Unknown node collects all the ID nodes and calculates the distance.
b. The unknown node arranges nodes ID by distance order.
c. Make decision based on DCP of tetrahedron
d. Make decision based on VRT
e. Use of Novel Centroid algorithm Z_{c1} = (G_cT*G_c)-1*G_cT*hc1)

3.2.4 DFPLE

DFPLE is proposed to minimize the mean error and computational price in estimating WSNs location [21]. Like CPE, DFPLE is based on a bounding box algorithm to estimate the candidate of location. DFPLE expands the bound for location estimation using three cases of beacon node positioning. DFPLE has dynamic number of bound points. First, some assumptions are required for wireless sensor networks:

a. There are N sensor nodes in the wireless sensor network.
b. Every sensor node has a unique ID.
c. Sensor nodes are deployed randomly.
d. There are M beacon nodes in the network, where $0 < M < N$.
e. Each beacon node is equipped with a GPS and, thus, knows its own location.
f. The other $(N - M)$ nodes are normal nodes that are unaware of their positions.
g. The transmission power of a beacon node is modulated by the variable radius method.

That is, the power level of beacon nodes can be modulated to high power level up to increase the communication range of beacon nodes to 2r, where r is the transmission radius of normal nodes. The DFPLE consists of four main phases on its operation: gathering beacon node location phase, estimating location, refining estimated location and error estimation. Each step of phases described as follows:

- ***Gathering Beacon Node Location***
  To gather information about other beacon nodes within communication range, beacon nodes must increase power to extend their communication range to *2r*. 2. The beacon nodes gather the ID and location information of neighboring beacon nodes by exchanging beacon frames.

- ***Location Estimation***
  1. Beacon nodes reduce power to their original level.
  2. Normal nodes record all neighbors within communication range.
  3. Neighboring beacon nodes provide other beacon node locations, which are collected in Phase I. When the beacon node is beyond the communication range of normal nodes, the neighboring beacon node that is farthest from the normal beacon node is considered the beacon node.
  4. The location of the normal node must meet one of the following three cases:
     - a. When the normal node is within communication range of a beacon node, the location of the beacon node is considered the most likely solution
     - b. When the normal node is within communication range of two beacon nodes, the midpoint of the intersection of their communication ranges is considered the most likely solution
     - c. When the normal node is within communication range of three beacon nodes, the Fermat point of the triangle which is formed by the intersection of the three circles in which the center of the circles are the beacon node locations is considered the most likely solution
- ***Determine FERMAT Point***

  The FERMAT point is point in ∆*PQR* that minimizes |*FP*| + |*FQ*| + |*FR*| (Figure 6). When all angles of △*PQR* are less than 120°, a unique Fermat point *F* lies inside the triangle such that and meet each other at mutual angles of 120°. The Fermat point is found as follows.
  - Construct a virtual equilateral triangle associated with each *PR*, *RQ*, and *QP*, designated ∆*PQ`R*, ∆*RP`Q*, and ∆*QR`P* respectively.
  - Construct lines *PP*`, *QQ*` and *RR*`. These are straight lines that connect the vertices of the triangle with the opposite vertices of the drawn virtual triangles.
  - Finally, *PP*`, *QQ*` and *RR*` intersect at the Fermat point, for which the sum of the distances from the point to the vertices of ∆*PQR* is minimal.
- ***Refining Estimated Location***

  ∆*PQR* needs to be shrunk to reduce the error in the estimated location. When ∆*PQR* is constructed from three neighboring beacon nodes, two vertices may have the same *x* or *y* coordinate. Figure 7 presents three constructions of ∆*PQR*—cases A, B, and C. Each case is treated with respect to refinement of the estimated location.

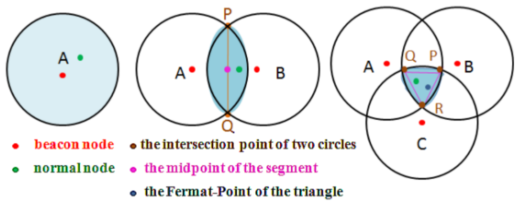

A
A
B
P
Q
A
B
C
Q
P
R
beacon node
normal node
the intersection point of two circles
the midpoint of the segment
the Fermat-Point of the triangle

Figure 17. DFPLE Operation

3.2.5 CPE

The convex position estimation (CPE) was proposed by proposed by Doherty *et al* [22]. The basic idea of the CPE is that if a sensor node can communicate with another sensor, its position is restricted by the connectivity constraints to be in some region relative to the other sensors. Many such connectivity or proximity constraints define the set of feasible sensor position in a WSN.

These constraints can be represented as Linear Matrix Inequalities (LMI-s). Once all the constraints in the network are expressed in this form, the LMI-s can be combined to form a single semi definite program. This is soled to produce a bounding region for each node, which Doherty *et al*., simplify to be a bounding box. If an unknown node can communicate with some neighbours anchor nodes, then there are connectivity constrains between the unknown node and its neighbours anchor nodes.

Since the location of the unknown node must within the overlapping region of the communication regions of these anchor nodes, the information such as locations and communication ranges of these nearby anchor nodes can be employed to estimate the location of the unknown node. The CPE algorithm define the estimative rectangle (ER) which bounds the overlapping region and regards the center of the rectangle as the estimative location of the unknown node.

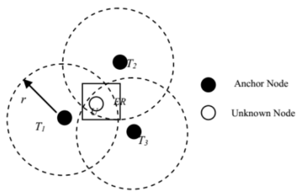


Figure 18. CPE Algorithm

The CPE algorithm is centralized localization scheme since each unknown sensor node sends the collected connectivity constraints back to a centralized controller. The centralized controller then estimates the location of every unknown node and flood the estimative location back to every unknown node. This central method makes the traffic-load in CPE heavy and the CPE algorithm scale poorly.

There are three main steps in the improved CEP localization algorithm [23]: Getting the information of the one-hop and two two-hop away neighboring anchor nodes of unknown nodes, getting the initial estimative location of unknown nodes and refining the initial location of unknown nodes.In the anchor exchange phase, every sensor node gathers the location information of anchor nodes, which is one-hop and two-hop away via anchor nodes two-hop flooding. By using two-hop flooding, every unknown node can gather the ID and the location information of its one-hop and two-hop neighboring anchor nodes

After finishing the anchor exchange phase, all the unknown nodes get the ID and the position of their one-hop and two-hop away anchor nodes. Then each unknown node computes its estimative rectangle (ER) as in CPE algorithm, and then uses the center of the estimative rectangle as the estimative location of the unknown nodes.In this phase, the initial estimative location obtained by utilizing estimative rectangle can be further refined by the information of neighbours two-hop away anchor nodes. The position the unknown node *N* can be calculated as:

$$P = P' + \sum_{i=1}^{m} \overrightarrow{A_\iota B_\iota}$$

where *P’* the initial estimative location of unknown node *N*, which is computed by using CPE method in the former subsection.

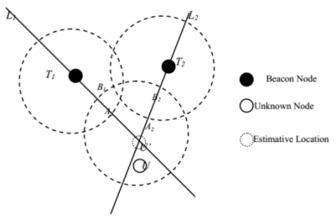


Figure 19. Improved CPE algorithm

3.2.6 3D DV-Hop

The DV-Hop localization algorithm as a distance-independent localization algorithm, is proposed by Dragos Niculescu from University of Lotto, USA [24]. The optimal principle is that the unknown node first calculates the minimum hop count of the beacon node, then estimates the average distance per hop, and then multiplies the average hop ratio by the minimum number of hops, finally obtains the estimated distance between the unknown node and the beacon node.

The measurement method or the maximum likelihood estimation method can be used to calculate the coordinates of the unknown node. Similar with the conventional DV-HOP algorithm, the 3D DV-Hop algorithm is also composed of three phases [24-26]. Firstly, the typical distance vector exchange protocol is used to obtain all the nodes in the network to get the number of hops from the beacon nodes.

In the second stage, after receiving the other beacon node position and the separated jump distance, the beacon node calculates the network average distance per hop, which is broadcast as a correction value to the network. And the correction value (average hop distance) *HopSize$_i$* of the beacon node (x_i, y_i, z_i) is expressed as follows:

$$\mathrm{HopSize}_i = \frac{\sum_{i \neq j} \sqrt{(\boldsymbol{x_i} - \boldsymbol{x_j})^{2} + (\boldsymbol{y_i} - \boldsymbol{y_j})^{2} + (\boldsymbol{z_i} - \boldsymbol{z_j})^{2}}}{\sum_{i \neq j} \boldsymbol{h_i}}$$

Where (x_i, y_i, z_i) (x_i, y_i, z_i), , are the coordinates of the beacon nodes i and j. h_i is the hop count of beacon i and all other beacon nodes. When the unknown node obtains a Euclidean distance between four or more beacon nodes, it can enter the third stage, i.e., calculate the node position. In the third stage, the position estimation is usually performed using a multilateral measurement method or a maximum likelihood estimation method. Assume that (x,y,z) is the coordinates of an unknown node U, which measures the distance of the coordinates of the n beacon nodes. The coordinates of the i-th beacon node are (x_i, y_i, z_i) and the distance from node U to beacon node i is *di*.

The 3D DV-Hop localization algorithm incorporates the constraints of the hop count and multiple collinearity thresholds in the general DV-Hop localization algorithm. The addition of the threshold improves the localization accuracy of the algorithm and reduces the computational complexity of the localization algorithm. The specific process of the algorithm is expressed in detail as follows:

1.Calculate the minimum number of hops for the unknown node and the beacon node within the defined hop count. The unknown node records the minimum number of hops received by the beacon node, while ignoring the larger number of hops from the same beacon node. When the unknown node receives the hop count value less than the threshold three_hops, the node increases the hop value by l and forwards it to the neighbor node. Otherwise, the packet is discarded

2. Calculate the distance between the unknown node and the beacon node within the defined hop count. Each beacon node uses the DV-Hop method to estimate the actual distance per hop based on the location information and the number of hops of the other beacon nodes. The unknown node records only the average distance per hop for the first average distance or the number of jumps per hop, so that the unknown node can receive the average distance per hop from the nearest beacon and calculate the hop count based on the number of beacon nodes within the hop distance

3. Use the node location method mentioned in the basic principle of node location to calculate its own position. First, the unknown node will calculate all its beacon nodes in four groups according to the set of MC, according to the set thre_mc excluded MC by discarding less than the reference point combinations with less than thre_mc, and then according to the multilateral measurement method to calculate the node coordinates, and finally take the average of all the results as the final position coordinates of the unknown node.

3.2.7 Enhanced APIT algorithm

The localization process of the APIT algorithm can be expressed as follows: The target node can get the set of beacon nodes which can communicate with it through the information transmission with the surrounding beacon nodes. Assuming there are n elements in the set and any three nodes are chosen to form a triangle, a total of C_n^3 triangles can be formed. It is judged that the position of the unknown node is inside the triangle or outside the triangle, and then the other three nodes are selected to determine the position of the unknown node until all triangles are exhausted.

Then, the overlapping area of all triangles can be calculated and further the area where the unknown node is located can be gradually reduced. Finally, the centroid of the overlapping area can be obtained as the estimated coordinates of the unknown node. The most important part of the APIT algorithm is to determine whether the unknown node is inside or outside the triangle. In this regard, the best-point-in-triangle-Φ (PIT) test algorithm can be used to determine whether the unknown node is inside the triangle. Figure 2 shows the specific PIT algorithm principle.

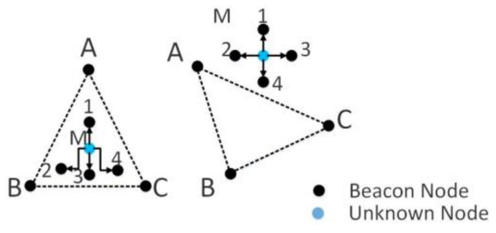


Figure 20. PIT diagram

A, B, and C are the three vertices of the triangle, and M is the unknown node that needs to be confirmed. Let the M point moves in any direction. When the M point is in the movement process and there is a point with the move to the M point, distances from the M point to A, B, C are increased or decreased at the same time, thus the point M is located outside the triangle; otherwise, M is located inside the triangle, which follows the PIT principle.

Enhanced APIT algorithm.

The improved APIT algorithm is similar with the APIT algorithm. First, we need to know the information of all the beacon nodes around the target node, which includes the position of the beacon node and the signal strength of the beacon node receiving the target node.

All beacon nodes that can communicate with the target node form a beacon node set. We arbitrarily select three beacon nodes to form a triangle to determine whether the target node is inside or outside the triangle. It is known that the three beacon nodes receive the signal strength of the target. When there is a point, the signal strength of the three beacon nodes received by the point is greater than or less than the signal strength received by the target node, then the target node is outside the triangle; otherwise, the target node is inside the triangle.

Select all the triangles that contain the target node and divide the triangle into four or six parts with vertical lines. If the triangle is acute triangle, it can be divided into six small intervals. And if the triangle is a rectangular triangle or obtuse angle triangle, it can be divided into four small intervals. By comparing the signal strength by the target node received by the three beacon nodes, it can determine which subrange the target node is located. Find the overlapping area of all the cells and find the center of mass position of the overlapping area, and the center position can be seen as the target node position.

3.2.8 Hybrid 3D Localization Algorithm

A hybrid localization algorithm based on APIT and DV-Hop is proposed by Miaochao Chen [25]. When the PIT is used to determine whether the unknown node is inside the triangular region composed of any three beacon nodes, the judgment condition is added under the original APIT algorithm condition.

The beacon nodes A, B, C and the unknown node M are obtained by the RSSI method, and then use the triangular cosine theorem to find the angle. If !AMC + !AMB + !BMC = 360 °, M is in the triangle; otherwise, it is judged that the M is outside the triangle.

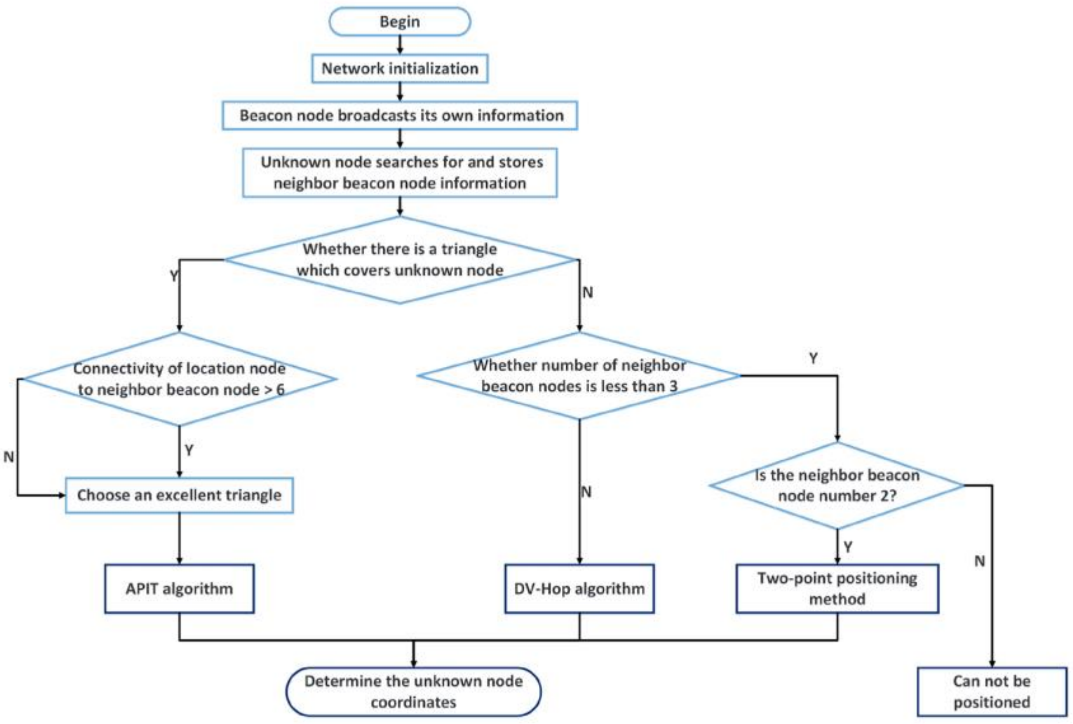

Begin
Network initialization
Beacon node broadcasts its own information
Unknown node searches for and stores neighbor beacon node information
Whether there is a triangle which covers unknown node
Y
N
Connectivity of location node to neighbor beacon node > 6
Whether number of neighbor beacon nodes is less than 3
Y
N
Y
Choose an excellent triangle
N
Is the neighbor beacon node number 2?
Y
N
APIT algorithm
DV-Hop algorithm
Two-point positioning method
Determine the unknown node coordinates
Can not be positioned

Figure 21. 3D localization algorithm based on APIT and DV-Hop

In the selection of fine triangles, remove the signal strength of the weak triangle, the selection principle is that: in the beacon node intensive, set a threshold P, and P is for the unknown node in the PIT point of all triangles 3 nodes of the signal strength and the average. In the sparse environment of the beacon node, and the neighbor beacon node is not less than 3, the DV-Hop algorithm is used to locate. Because it calculates the minimum hop count of the unknown node and the beacon node by the distance vector routing method and then calculates the average distance of each hop.

The product of the average distance between each hop and the minimum hop count is used as the estimated distance between the unknown node and the beacon node. When there are three beacon nodes, the coordinates of the unknown node are calculated using the trilateral method or the maximum likelihood estimation method. With only two neighbors of beacon nodes, the two-point localization method should be applied for node localization. The two-point localization method can measure the distance of unknown nodes and two beacon nodes according to RSSI, and then the coordinates of the unknown node can be obtained according to the coordinates of these two beacons.

3.2.9 3D-TDOA Fictitious Point Method

TDOA systems are based on difference time measurements between the signal arrival to different nodes or sensors in a network [26]. These measurements can be converted to difference of distances by multiplying these times by speed emission of the radioelectric waves (c). This leads in Euclidean Geometry to the next equation:

$$R_{ij} = d_{ij} = d_{Ii} - d_{Ij} =$$

$$\sqrt{(x_I - x_i)^2 + (y_I - y_i)^2 + (z_I - z_i)^2} - \sqrt{(x_I - x_i)^2 + (y_I - y_i)^2 + (z_I - z_i)^2} + h(0, \sigma) =$$

$$ct_{ij} + h(0, \sigma)$$

where dIj is the distance difference between receivers i and j—which is the result of multiplying the actual time difference of arrival (tij) and adding a white noise, h(0, σ), that considers atmospheric instabilities and time error measurements.

This noise is related to signal transmission and measurement of times, which cannot be controlled by TDOA algorithms and so is not considered in this paper. In addition, (xI, yI, zI) are space coordinates of the vehicle that are being positioned and (xi, yi, zi), xj, yj, zj are coordinates of the nodes i and j, respectively, which receive the positioning signal. These equations correspond with hyperboloids that cannot be solved in an analytic direct process.

Non-linear equations of hyperboloids must be treated in order to address the TDOA problem resolution. Generally, two main methodologies have been considered: those based on hyperboloids intersection properties with closed-form solutions, and those based on numerical methods, which offer a progressive reduction on the error gradient derivation in successive approximations leading to the final solution. However, both of them share the qualification that a univocal TDOA problem resolution must use at least five different sensors.

3.2.9.a Intersection of hyperboloids.

The hyperboloid intersections can always be contained in a plane [27]. This process increases the freedom to the problem by one degree, since a number of n receivers generate a number of (n-1) independent hyperboloid equations and (n-2) independent intersection planes are obtained using this methodology. That means that to solve the 3D TDOA problem linearly, where three planes are needed, we still have to use five different receivers.

Nevertheless, the fact that the intersection of two different hyperboloids is contained in a plane makes the process of obtaining this plane equation independent from the original hyperboloid equations. As consequence, the intersection of two planes (four nodes) resulting in a line of possible vehicle localizations can be verified in any hyperboloid to finally get the two solutions that are achieved in TDOA problems with four beacons (i, j, k, l). This methodology leads to two different solutions that for LPS cannot be discarded by any assumable criterion.

3.2.9.b Taylor approximation

The other method would be based on applying a Taylor approximation truncated on first order to linearize the equations and allow a real-time solution to the problem. In this way, a point with enough proximity to the final solution (x0, y0, z0) from which a process of sequential iterations will be started is selected. These steps will finally allow the vehicle localization to be obtained through a matrix where the range differences are considered as follows:

$$R_{ij} = ct_{ij} = R_{ij0} + \frac{\partial R_{ij}}{\partial x}\Delta x + \frac{\partial R_{ij}}{\partial y}\Delta y + \frac{\partial R_{ij}}{\partial z}\Delta z$$

where R_{ij} is the value of the distance difference in the approximation point, and $\frac{\partial R_{ij}}{\partial x}, \frac{\partial R_{ij}}{\partial y}, \frac{\partial R_{ij}}{\partial z}$ are partial derivatives of the range differences, particularized for the values of the approximation point. Applying this very same process to the other two nodes k and l with reference to the node i, Rik and Ril can be estimated. This leads to the following matrix system:

$$\Delta R = \begin{bmatrix} \frac{\partial R_{ij}}{\partial x} & \frac{\partial R_{ij}}{\partial y} & \frac{\partial R_{ij}}{\partial z} \\ \frac{\partial R_{il}}{\partial x} & \frac{\partial R_{il}}{\partial y} & \frac{\partial R_{il}}{\partial z} \\ \frac{\partial R_{ik}}{\partial x} & \frac{\partial R_{ik}}{\partial y} & \frac{\partial R_{ik}}{\partial z} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}$$

where ΔR is the range differences matrix, H is the partial derivative matrix (commonly known as visibility matrix) and P is the position variance matrix. Therefore, we can express the matrix system as follows:

$$H\Delta P = \Delta R$$

This equation is usually solved through the least squares method [18], as described below:

$$\Delta P = (H^t H)^{-1} H^t \Delta R = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}$$

Regarding the resolution of the TDOA problem, four receiving sensors do not always guarantee the convergence of the method and, if produced, this can affect any of the two possible solutions (which prevents us from knowing whether the position calculation is correct). However, in contrast with the former method, the calculation of the position now guarantees a single solution instead of two possible answers.

3.2.9.c Fictitious Point Method

Of all the methods proposed so far, it is not possible to conclude whether the TDOA [28] System can be applied to LPS systems with four nodes with enough confidence to guarantee the correct calculation of the position. Nevertheless, it is possible to affirm that successive approximation methods do guarantee convergence—if produced—towards one of the possible of the solutions. This means that if there were any way to ensure that the convergence occur toward the correct solution, the method would allow the problem with to be solved with four sensors.

In a scenario where the process is convergent and highly dependent upon the initial point of the iterations, it is safe to say that when this initial point is close enough to the solution (i.e., the previous solution of the vehicle), the convergence should always take place toward the correct solution. To prove this statement, the behavior of any point located at a plane containing the two possible solutions is going to be proven for the TDOA problem. The solution has been calculated by applying the successive approximation method to these initial points.

3.2.10 3D-TDOA CHAN Method

It is a proposed algorithm from Zhang Jian-wu, Yu Cheng-lei, Tang bin and Ji Ying-ying [65]. Mobile positioning was achieved through detection signal of MS arrival time difference of two BSs in The TDOA technology, which greatly reduces the time synchronization requirements. In this algorithm, a hyperbola could be got with the focus of two BS and the focal length of the distance difference between MS to two BSs. Thus, it could measure the two-dimensional (2D) coordinates of MS by the intersection of three hyperbolas

Several typical location algorithms based on TDOA are Fang algorithm, Chan algorithm, Taylor series expansion method and so on. With high positioning accuracy, the location technology of TDOA could reduce the system error, and the performance of TDOA is well in the non line-of-sight (NLOS) environment. However, comparing with TOA, the algorithm of TDOA is complex. Usually, positioning algorithm is based on two dimensional coordinates. In Practical application, not only two-dimensional coordinates of MS should be measured, but also height coordinate should be got. Thus, it is most important to measure the 3-dimensional coordinates of MS. Especially in the downtown with numerous high-rise buildings or the fluctuating level of the mountain environment, this demand even more urgent.

Whether TOA or TDOA wireless location technology, accurate ranging is the first step to obtain the precise positioning. There are two way of traditional ranging of TOA, one is Two Way Ranging (TWR), another is One Way Ranging (OWR). TWR: If there is no common clock between nodes, the time between transmit node and receive node can be used to estimate the distance between two nodes. As shown in figure 1, node A sends a packet to node B at T0. After node B receives the packet, it returns a packet to node A at once. Node A receives the return packet at T1.

The distance between node A and node B may be calculated. OWR: If there is common clock between nodes, we could use the OWR to measure the distance between two nodes. Time between two nodes could directly be measured by this method. Node A sends a packet to node B at T0, and node A receives the packet at T1 . The distance between node A and node B may be calculated. To unknown node coordinates can be calculated with the usage of hyperbolas and linear algebra, more information in the implementation of the algorithm are provided in the next sessions.

3.2.11 TOA/RSSI Direct Location Method

A proposed algorithm from Mohamed Khalaf-Allah [66]. The positioning algorithms include the analytical method, the least-squares method, Taylor series method, the approximate maximum likelihood method, two-stage maximum likelihood method and the genetic algorithm. The described analytical (direct) method has three possibilities for the solution of x, i.e. one solution, two solutions and no solution, unlike the developed method, which yields a direct, exact and unique solution. This is due to using exactly four TOA measurements rather than three TOA measurements and thus any ambiguities can be resolved. The algorithm is advantageous in terms of implementation simplicity and computational cost. Therefore, it can be easily implemented in many low-power and low-cost wireless applications, e.g. 3D sensor networks. It uses four TOA measurements with four stations to avoid object location ambiguities, which is a problem associated with using three TOA measurements with three stations. Another advantage of this direct method is that it needs exactly four TOA measurements to compute 3D position solutions, where in many situations more than four measurements are not available or the availability of more than four measurements is not important for the accuracy requirements of the application at hand. The 3D position algorithm delivers position solutions that are dependent only on the given measurements and information, in case the interest is only in a single coordinate, e.g. the vertical component. More information for the implementation of the algorithm are presented to the next sessions.

3.2.12 Hybrid 3D-TOA/TDOA

A hybrid TOA/TDOA lateration algorithm has been proposed from Yaro Abdulmalik Shehu, Muazu Musa, Sani Salisu and Abdulrazaq Abdulaziz [67]. Passive multilateration (MLAT) surveillance system estimate aircraft location in two steps. The first step involves estimating the time difference of arrival (TDOA) of the signal at antenna pairs. The second step involves using the estimated TDOA measurements from the first stage as input to a position estimation (PE) algorithm known as lateration. 2D or 3D aircraft PE depends on the number of antennas deployed. For a 3D aircraft PE, a minimum of 4 antennas are required. Several techniques for estimating TDOA have been reported in literatures but the classical approach use in air traffic surveillance is the TOA approach. TDOA estimation using the TOA approach involves a pair wise difference of TOA measurements of the signal estimated at each antenna. TOA of the signal is the time taken for the transmitted signal from the aircraft to be detected at any of the antennas. More information are presented to the next sections.

3.2.13 3D-RSSI/TOA Multilateration

This method follows the concept of calculating the distances of an unknow node from three base nodes with the RSSI/TOA method, which those 3 distances represents three radii of three spheres with centers the coordinates of the base nodes.

3.2.14 Localization Algorithms Summary

Table 3. Localization Algorithms Summary

Name	Nodes	Publication	Simulation and Information
FP-MPP-APIT	***4***	***March 9, 2018 [17]***	MATLAB 2014a was utilized as a simulation platform: Space:100m x100m x100m.Total Nodes :200.Beacon Nodes :20.Range: R=15. The proposed algorithm was compared with PB-APIT-3D,FM-APIT-3D and DFPLE algorithm and showed that has a smaller localization error. Minimum 5 base nodes
3D-IDCP	***5***	***7 June 2015 [18]***	MATLAB 2009a.Three-dimensional area of 100 m × 100 m × 100 m. Communication radius *R* is set 40 meters. Experiments are carried out 50 times. Each time sensor nodes are randomly deployed. The simulation experiments have verified the effect of the proposed algorithm in positioning accuracy, positioning coverage, and proportion of bad nodes. Minimum 5 base nodes
Novel Centroid	***5***	***November 2008 [20]***	Size of 100 m x 100 m x 100 m. centroid location is robust under the effect of the irregularity of the radio pattern. The reason is that the centroid algorithm does not depend on hop-count and hopsize that the effect of degree of irregularity（ DOI） is abated by the aggregation of beaconed information.3.The sensor nodes have the same maximum radio range *R,* which is used for normalization only.100 sensor nodes 4. The simulation results are averaged over 100 network instance. The proposed algorithm can improve location accuracy than the conventional centroid localization algorithm. Minimum 5 base nodes
DFPLE	***5***	***2011 [21]***	MATLAB A 2-D square area (5r5*r* and 10*r*10*r*) in which sensor nodes were randomly deployed. 200 nodes and radio range of 1.5*r* The DFPLE strategy was compared with the Convex Position Estimation (CPE). Simulation results demonstrate that the DFPLE algorithm for estimating sensor positions is more accurate than existing algorithms and improves upon conventional bounding box strategies. Minimum 5 base nodes
CPE (improved)	***5***	***2015 [22][23]***	MATLAB, Area of 200*m*× 200*m*.The sensor nodes are distributed randomly in this region. Each sensor node has the same communication range 20*m*. With the number of sensor nodes is 200 and 300. The improved CPE localization algorithm outperforms the classic CPE and does not increase the hardware cost of sensor nodes. Minimum 5 base nodes. Minimum 5 base nodes
3D DV-Hop	***5***	***25 January 2017 [24]***	MATLAB .100x100x100 m^3 3D space. The obstacle is a cube with the length of 14m, distributed in the 48m<x,y,z<62m2. The total number of nodes is 200, including 50 beacon nodes and 150 unknown nodes. The communication radius of anchor node is 30m. The proposed algorithm is significantly superior to the improved DV-Hop localization algorithm and the traditional DV-Hop algorithm. Minimum 5 base nodes
Enhanced APIT algorithm	***5***	***2015 [25]***	MATLAB Wireless sensor network coverage: 1000m % 1000m two-dimensional plane.200 sensor nodes, Radius is 200m, Minimum 5 base nodes
Hybrid 3D Localization	***5***	***2017 [26]***	MATLAB 1. Wireless sensor network coverage: 1000m % 1000m two-dimensional plane. 2. The network layout of 200 sensor nodes, Communication radius is 200m, Simulation results show that the proposed hybrid algorithm can effectively improve the localization accuracy of beacon.
3D-TDOA Fictitious Point	***4***	***2019***	Field: 1000 × 400 × 100 m, described with a spatial discretization of 100 m in x coordinate, 50 m in the y coordinate and 10 m in the z coordinate. Each of the discretization points represents a real solution to the 3D TDOA system of study. Results showed that he four-sensor TDOA problem can be solved with only four sensors within a confidence interval defined through the convergence radius. Minimum 4 base nodes
3D-RSSI/ TOA Multilateration	***“3” - 4***	***-***	Simulation is presented in the section 7 of this paper alongside with information of the implementation of the algorithm in section 6
3D-TDOA CHAN	***5***	***2008 [65]***	Two simulations were performed with a numerical example. The increase of the number of available BSs, the positioning accuracy of Chan algorithm has some increased in the environment with Gaussian noise. When the value of NLOS is from 0 to 50, the performance improvement caused by increasing the number of BSs is not very significant. The reason is that with little value of NLOS, positioning accuracy is mainly determined by system error. However, with the increasing of the value of NLOS, the trend of positioning accuracy improvement caused by increasing the number of BSs is great. with the increase of the height of MS, the positioning accuracy of Chan algorithm has improved.
3D TOA/RSSI Direct Location	***4***	***2014 [66]***	Simulation was performed with a numerical example. Four fixed transmitters were located at (0,0,0), (10,0,5), (10,10,0) and (0,10,5). The error-free range measurements were respectively 75 , 50 , 75 and 50 , where all coordinates and range measurements are in meters.
Hybrid TOA/TDOA	***4***	***2016 [67]***	The performance in PE of the developed algorithm is compared with the two fix reference TDOA lateration using Monte Carlo simulation for some selected aircraft locations and TOA error standard deviation range of 0 meters to 2 meters. Simulation results shows that the performance comparison between the two lateration algorithms depends on the location of the aircraft.

4. INTRODUCING SECURITY IN GEO-LOCATION

4.1 Attacks in Localization

Secure localization of unknown nodes in a Wireless Sensor Network (WSN) is an important research subject. When WSNs are deployed in hostile environments, many attacks happen, e.g., wormhole, sinkhole and sybil attacks. Two issues about unknown nodes' secure localization need to be considered. First, the attackers may disguise as or attack the unknown and anchor nodes to interfere with localization process. Second, the attackers may forge, modify or replay localization information to make the estimated positions incorrect [28].

The location information of the sensor node performs a critical role for numerous applications in wireless sensor networks (WSNs) such as environment monitoring, target tracking, and automatic surveillance. It also helps some fundamental techniques in sensor networks (e.g., geographical routing protocol and topology control) to be aware of where the messages are located. Driven by those demands, earlier research efforts have resulted in many localization schemes, with most assuming the sensors are deployed in a benign scenario. But when the sensor nodes are deployed in malicious environments, it is prone to different forms of threats and risks. A simple malicious attack can disturb the accurate position estimating and even make the entire network functioning improperly. Usually, the localization process can be divided into two steps information acquisition and position determination.

4.1.1.a Information acquisition

Roughly speaking, existing localization schemes of WSNs are classified into two categories: range-based schemes and range-free schemes. For range-based localization schemes, the distance or angle information is measured by RSSI etc. For range-free localization schemes, the localization is realized based on network connectivity or other information.

4.1.1.b Position determination

Location determination schemes have two categories: terminal-based schemes and infrastructure-based schemes. In terminal-based schemes, the unknown node localizes itself, the position of an unknown node can simply be computed by trilateration etc. In infrastructure-based schemes, references nodes including trusted neighbor nodes, mainly anchor nodes to localize the unknown node.

Localization process can be attacked in different ways. Researchers have addressed a set of known attacks. The known attacks can be divided into two categories: external and internal attacks. The adversary is external if it is outside the WSN and implements malicious behaviors without right cryptographic key. Otherwise, the adversary is internal, in which case the adversary controls one or more fraudulent nodes. The attacks can also be classified into three categories Attacks on nodes, Attacks on information and Dos Attacks [29] [30].

4.1.1 Attacks on Nodes

An attacker is an external node which intrudes into the WSN. A compromised node is a normal node (an unknown or an anchor node) in the WSN compromised by the attacker. Attacks on nodes are listed as follows:

4.1.1.a Compromise

Node compromise is the most fundamental attack in WSN that leads to other kinds of attacks. It occurs when an attacker gains control of a node in the WSN. Normally, compromised nodes can be obtained by the following methods: attackers capture normal nodes and reprogram them attackers deploy nodes with larger computing resources such as laptops to attack normal nodes. With compromised node, an attacker can alter the node to listen information in the WSN, revoke legitimate nodes, input malicious data, and cause internal attacks, e.g. DoS attack.

4.1.1.b Replication

If an adversary manages to capture a node and extract the authentication/encryption keys, it can produce many replicas having the same identity (ID) from the captured node and integrate them into the WSN at chosen locations, which is called the node replication attack. Since the credentials of replicas are all the clones from the captured nodes, the replicas can be considered as legitimate members of the network. It is always assumed that the adversary cannot create new IDs for replicated nodes, since otherwise the attackers will have to create the corresponding security information (keys, codes, etc.), which is very difficult and even infeasible in most cases. Once the adversary replicates one or more sensor nodes, it can execute the malicious operations.

4.1.1.c Impersonation

An impersonation attack is an attack in which an adversary successfully assumes the identity of one of the legitimate parties in a system or in a communications protocol. One form of node impersonation attack is the Invisible Node attack, and the other one is the Stolen Identity attack. The Invisible Node attack: Malicious node M simply stands between two nodes A and B that are not in direct range. The invisible node M silently repeats the communication between nodes A and B, which misleadingly assume that nodes A and B communicate directly. In this way, the malicious node succeeds in impersonating node A to node B and vice versa.

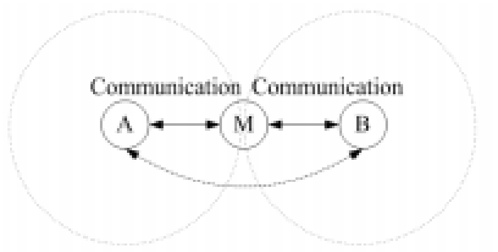


Figure 22. The invisible node attacks

The Stolen Identity attack: The malicious node M succeeds in stealing all the authentication credentials from a legitimate node A, such as the certified signature keys. If the malicious node outraces the legitimate node in updating the stolen credentials, then the credentials of the legitimate node will not be valid anymore. Thus, only the malicious node will be able to communicate with node B. This kind of attack is not just a matter of stealing a nodes identity, but also a matter of abusing the trust relationships that other parties may have had established with the legitimate node.

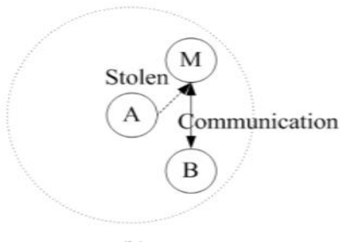


Figure 23. The stolen identity attacks

4.1.1.d Sybil attack

In this attack, a single node i.e. a malicious node will appear to be a set of nodes and will send incorrect information to a node in the network. The incorrect information can be a variety of things, including position of nodes, signal strengths, making up nodes that do not exist. Authentication and encryption techniques can prevent an outsider to launch a Sybil attack on the sensor network. However, an insider cannot be prevented from participating in the network, but he should only be able to do so using the identities of the nodes he has compromised. Public key cryptography can prevent such an insider attack, but it is too expensive to be used in the resource constrained sensor network.

4.1.1.e Wormhole attack

In a wormhole attack, an attacker records a packet or individual bits of a packet at one location in the network. Then, it tunnels the packet (possibly selectively) to another location and replays it. The tunnel can be established in different ways, for example, through an out-of-band channel, packet encapsulation, high-powered transmission, packet relay and protocol deviations. In localization process, the attack may tunnel totally different and erroneous localization information.

One node in the network (sender) sends a message to another node in the network (receiver node). Then the receiving node attempts to send the message to its neighbours. The neighbouring nodes think the message was sent from the sender node (which is usually out of range), so they attempt to send the message to the originating node, but it never arrives since it is too far away.

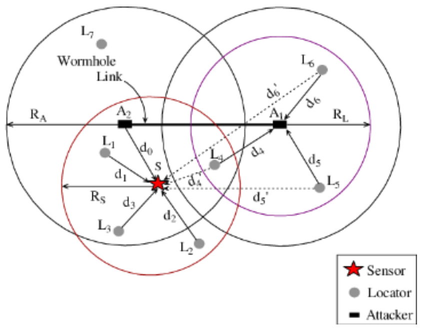


Figure 24. Wormhole attack

Wormhole attack is a significant threat to wireless sensor networks, because, this sort of attack does not require compromising a sensor in the network rather, it could be performed even at the initial phase when the sensors start to discover neighbouring information. Wormhole attacks are difficult to counter because routing information supplied by a node is difficult to verify.

4.1.2 Attacks on Information

In the localization systems, unknown nodes always use the localization information of anchor nodes to localize themselves. The target of malicious nodes is usually to make localization information incorrect. Attacks on information are listed as follows:

4.1.2.a Forgery

Forgery attack is the malicious node sends misleading information in the localization systems. For example, in the active system, the malicious node pretends to be an anchor node to voluntarily send localization information. In the passive system the malicious node pretends to be an unknown node to be localized.

4.1.2.b Alteration

Alteration attack is the most direct attack. This attack targets the information exchanged between an unknown and an anchor node. Adversaries may directly alter the coordinates, time or the number of hops and increase the localization error of unknown nodes. For example, in Collaborative Collusion [], all malicious node can collaborate with each other to alter the information they receive or replay.

4.1.2.c Interference

Interference attack is the malicious node interferes with the signal measurements. For example, in range-based localization systems, malicious nodes may place obstacles between signal sender and receiver to prolong transmission time

4.1.2.d Replay

Replay attack is the most common or simple attack, especially when the capability and resources of the adversary are limited. In this attack, the malicious node congests the information transmission between sender and receiver, then replays the outdated information. Using the outdated information, the unknown nodes calculate inaccurate positions. Unlike other attacks, replay attack can destroy the whole network with one node.

4.1.2.e Selective forwarding

In selective forwarding attack the malicious node behaves like black hole and refuses to forward sensitive messages and simply drops them, ensuring that they are not propagated any further.

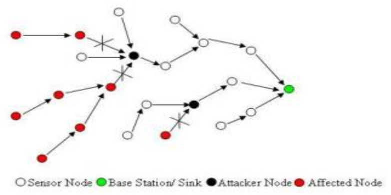


Figure 25. Selective forwarding

The selective forwarding attack is difficult to detect. First, to avoid raising suspicions, an adversary selectively drops packets instead of dropping every packet. In addition, there are many reasons result in packet dropout, e.g., unreliable wireless communications, sensor nodes go into sleep state to save power.

4.1.3 Denial of Service Attacks (DoS)

4.1.3.a Jamming

Jamming is a DOS attack at physical layer. Jamming interferes with the radio frequencies that a network’s nodes are using. A jamming source may either be powerful enough to disrupt the entire network or less powerful and only able to disrupt a smaller portion of the network

4.1.3.a Tampering

Another DOS attack in physical layer is tampering. By physical access an attacker can extract sensitive information such as cryptographic keys or other data on the node. A compromised node creates, which the attacker controls by altering or replacing node. Vulnerability of this attack is logical. A defence to this attack involves tamper-proofing the node’s physical package.

4.1.3.b Collisions

Collision is a DOS attack in the data link layer. When two nodes attempt to transmit on the same frequency simultaneously a collision occurs. A change will likely to occur in the data portion when packets collide and causing a checksum mismatch at the receiving end. The packet will then be discarded as invalid. An adversary may strategically cause collisions in specific packets such as ACK control messages. Error-correcting codes use to defend against collisions.

4.1.3.c Exhaustion

It is another type of DOS attack in link layer. An attacker can use repeated collisions to cause resource exhaustion. For example, a native link-layer implementation may continuously attempt to retransmit the corrupted packets. The energy reserves of the transmitting node unless these hopeless retransmissions are discovered or prevented. Applying rate limits to the MAC admission control is a possible solution of exhaustion.

4.1.3.d Unfairness

Unfairness is a weak of a DOS attack in link layer. An attacker may cause unfairness in a network by using the above link- layer attacks. Instead of preventing access to a service outright, an attacker can degrade it in order to gain advantage such as causing other nodes in a real time MAC protocol to miss their transmission deadline.

4.1.3.e Selective Forwarding Attack

Multi-hop mode of communication is commonly preferred in WSN data gathering protocols. An assumption made in multi-hop networks is that all nodes in the network will accurately forward received messages. Selective forwarding attack is a situation when certain nodes do not forward many of the messages they receive. In this attack, malicious nodes may refuse to forward certain messages and simply drop them, ensuring that they are not propagated any further.

4.1.3.f Flooding

Flooding attack on localization is the malicious node broadcasts large quantities of useless data packets to all nodes in its communication range. The common characteristic of flooding attack is to exhaust the available network communication bandwidth so that the other nodes cannot communicate with each other. Moreover, the sender and receiver are busy to send or receive the excessive packets from the attacker and consume a lot of network resources.

4.1.3.g Desynchronization

Disruption of an existing connection is desynchronization. For example, an attacker may repeatedly spoof messages to an end host, causing that host to request the retransmission of missed frames. With proper timing, an attacker may degrade or even prevent the ability of the end hosts to successfully exchange data. A possible solution to this type of attack is to require authentication of all packets communicated between hosts. The authentication method would be secure as an attacker will be unable to send the spoofed messages to the end hosts.

4.1.4 Localization attacks summary

All the previously mentioned security threats serve one common purpose that is to compromise the integrity of the network they attack. In the past, focus has not been on the security of WSNs, but with the various threats arising and the importance of data confidentiality, security has become a major issue. Although some solutions have already been proposed, there is no single solution to protect against every threat.

Table 4. Localization attack summary

Attack Name	Attack Behavior
Compromise	Alter the node to listen information
Replication	Node replication
Impersonation	Node impersonation
Sybil attack	Possessing multiple identities
Wormhole attack	Shortening the distance to make a fast routing path
Forgery	Sends misleading information
Alteration	Alter node coordinates
Interference	Interference signal measurements
Replay	Replays the outdated information
Flooding	Establishing false connections
Selective forwarding	Selectively forward packets
Stealing	Signal eavesdropping and tampering
Jamming	Sending jamming signal in the working frequency range
Collision	Repetition of messages
Exhaustion	Sending of unnecessary message
Unfairness	Explicitly taking the control of the channel
Dos Attacks	Exhaustion of energy of the unknown nodes
Sinkhole	Maliciously tamper with routing
Tampering	Tampering localization beacons
Insider attack	Compromised anchor nodes may provide false information
Range change attack	Changing the range or Angle of Arrival (AoA)
False beacon location attack	Compromising a beacon and then he can make the beacon broadcast false location
False reported location attack	Malicious node reports false
Desynchronization	Disruption of an existing connection

4.2 Secure Geo-location Schemes

Localization algorithms are an important and challenging topic in Wireless Sensor Networks (WSNs), especially for the applications requiring the accurate position of the sensed information. Various algorithms have been proposed to obtain the location of sensor nodes. However, most of existing location algorithms assumes a non-adversarial environment.

The position estimation accuracy decreases drastically when some of the sensor nodes are compromised. In this section we present a variety of localization algorithms that ensure user's authentication, security and data safety and resists malicious attacks. Safety in geo localization systems can be achieved in many ways, for instance by using the coordinates of the unknown node as a verifier, cryptography or by external frameworks usage such as a TPM. Below we present some representative security algorithms and schemes in geo localization systems.

4.2.1 Location-Dependent data Encryption Algorithm (LDEA)

This algorithm is proposed by Hsien-Chou Liao, Yun-Hsiang Chao and Chia-Yi Hs [32]. It utilizes the latitude/longitude coordinate as the key for data encryption. When a target is determined for data encryption the cipher text can only be decrypted at the expected position.

a) The purpose of LDEA is mainly to incorporate the latitude/longitude coordinate in the data encryption and thus to restrict the location of data decryption.
b) A toleration distance (TD) is designed to overcome the inaccuracy and inconsistent problem of GPS receiver.
c) When the target coordinate and TD is given by the sender (information system or mobile user), an LDEA-key is generated from latitude/longitude coordinate and TD. The random-key generator issues a session key, called R-key.
d) Then, the final-key for encrypting the plaintext is generated by exclusive-or R-key with LDEA-key. The final-key can be used for the symmetric encrypt algorithm, such as DES, AES. KU_r and KR_r is the public and private keys generated on the receiver side. KU_r is transmitted to the sender first.
e) Then, TD and R-key is transmitted via asymmetric encryption algorithm. When the receiver gets the TD and R-key, the LDEA-key can be generated from TD and the coordinate acquired from GPS receiver. The final-key can be generated by exclusive-or R-key with LDEA-key.
f) If the acquired coordinate is matched with the target coordinate within the range of TD, the cipher text can be decrypted back to the original plaintext. Otherwise, the result is indiscriminate and meaningless.

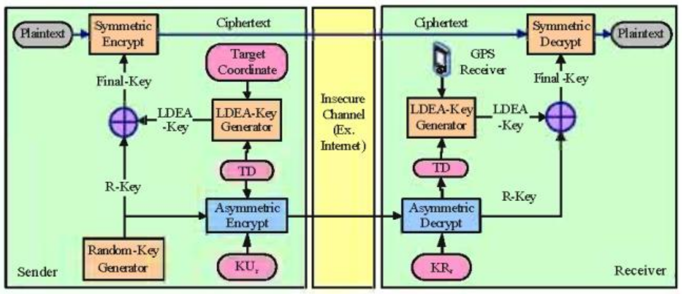


Figure 26. LDEA Process

4.2.2 Mutual Authentication Insider Node Validation

This algorithm is proposed by Gulshan Kumar, Mritunjay Kumar Rai, Hye-jin Kim and Rahul Saha [33]. The main concept of this algorithm is that the main consideration of location discovery is a set of special nodes known as anchor nodes, which are resource privileged having more storage and computational capacity. Using the location of anchor nodes, other unknown nodes compute their location in different ways.

Therefore, it is critical that malicious anchor nodes need to be prevented from providing false location information as the unknown nodes completely depend on the anchor nodes for computing their own location. The proposed algorithm considers only the anchor nodes, unknown nodes, and Base Station where anchor nodes and unknown nodes are deployed randomly. The anchors are having a variable range of transmission with an average transmission range Ravg given as:

$$\mathrm{Ravg} = \frac{\sum_{e \in E} \psi(|e|)}{m}$$

where m is the number of anchor nodes in the network, e is an edge between two nodes, E is the set of the edges in the network, and $\psi(|e|)$ is the weighing function of a connection between an anchor node and an unknown node and interpreted as $\psi(|e|) \sim |e|\alpha$, $2 \leq \alpha \leq 4$. The algorithm starts with an initialization phase that deals with distribution of certificates by the BS. After the distribution of the certificates, distance estimation phase starts among the anchor nodes and the unknown nodes. Once the distances are estimated, the BS is able to localize the unknown nodes applying MMSE method.

4.2.2.a Initialization Phase.

Base Station (BS) provides the identity for all anchor nodes and unknown nodes as IDa_j and IDu_i where aj is an anchor node and u_i is an unknown node. BS also provides certificates for each anchor node and unknown node as Certa_j and Certu_i.

4.2.2.b Distance Estimation Phase.

The anchor node a_j sends a random nonce κ, along with the certificate Certa_j to all the one-hop neighborhood unknown nodes u_i in the range R_{avg} and starts the timer on. When the unknown nodes receive the message, verify the certificate using the public key BSK+ given by BS. As, only legitimate anchor nodes are having the certificate to provide, by verifying the certificates, the authentication of the anchor nodes can be proved. Then, the unknown nodes u_i response back to the anchor node aj with the same nonce κ, time duration between of receiving the last bit of message sent by anchor node and transmitting the first bit of message to the anchor node, given as time_{procu} encrypted with anchor node's public key Ka_{j+} along with its own certificate.

$$a_j \rightarrow u_i : \varkappa, \mathrm{Cert}u_i,$$
$$u_i \rightarrow a_j : [\varkappa, \mathrm{timeproc}u]\, K_{aj+}, \mathrm{Cert}u_i.$$

When a_j sends message to u_i, it waits for a bounded time value $t_{\text{retransmit}}$ to retransmit the message if no response starts arriving to the anchor in that bounded time. This value is precomputed at the starting of the network deployment assuming all the favourable conditions of the network environment with a noise effect of Δt and given as

$$t_{\text{retransmit}} = \text{time}_{\text{normal}} + \Delta t,$$

where $\text{time}_{\text{normal}}$ is the normal time duration of getting a response back from the unknown node. When the anchor node receives the response back from the unknown nodes, it decrypts the message using its own private key Kaj-, verifies the certificate of the unknown nodes, stops the timer, and calculates the signal propagation time as) where time prop is the signal propagation time, time$j$ is the timer interval at the anchor side, and time proc$a$ is the time duration between receiving the first bit of the response and last bit of the response. Once the propagation time is calculated, the estimated distance between anchor node aj and unknown node ui is calculated as:

$$d_{ui}^{aj} = c * \ time_{prop}$$

where c is the speed of light. Once the anchor node calculates this estimated distance, it is then forwarded to the BS encrypted with the public key of BS and along with the anchor node's certificate.

$$a_j \rightarrow \text{BS: } [d_{ui}^{aj}]_{\text{BS}K}, \text{ Cert}_{aj}.$$

After receiving the message from the anchor nodes, BS decrypts the message with is private key and gets the estimated distances. Finally, it uses Minimum Mean Square Error (MMSE) to estimate the location of an unknown node (xu_i, yu_i). The relative mobility between an unknown node ui and anchor node aj at a given time t is given by:

$$RM_t^{a,u} = \text{d}_{\text{a,ut}} - \text{d}_{\text{a,ut-1}}$$

where $RM_t^{a,u}$ is positive if node ui is moving away from aj and negative if ui is coming closer to aj.

4.2.2.c Handling Distance Estimation Error.

Distance estimations in a wireless environment are very common to have error due to the noise or delay in the medium. Assume that the estimation error is $\epsilon \in [-\epsilon\text{max}, \epsilon\text{max}]$, where ϵmax is a system parameter and given as $0 \leq \epsilon\text{max} \leq 1$. Therefore, the estimated distance can be given as:

$$d_{ui}^{aj} \in [\text{ true } d_{ui}^{aj} \text{ x } (1 - \varepsilon_{\max}), \text{true } d_{ui}^{aj} \text{ x } (1 + \varepsilon_{\max})]$$

where true $\boldsymbol{d_{ui}^{aj}}$ is the true distance between $a\ j$ and ui and can be calculated by applying Euclidean method.

4.2.2.d Simulation Results

The framework was compared with the three recent algorithms: (1) CSLT, (2) MPA and (3) AWS. The localization ratio is defined as the percentage of successful location estimation of unknown nodes. The results showed that the proposed algorithm performed better as compared to others. In the simulation, the ratio of malicious nodes varied from 5% to 30% with increments of 5%. Simulation results showed that the relative error percentage of location estimation increases with the increasing number of malicious nodes. However, the proposed algorithm proved its efficiency in location estimation accuracy

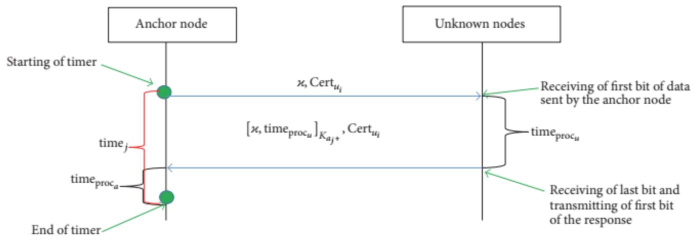

Anchor node
Unknown nodes
Starting of timer
$\varkappa$, Cert$_{u_i}$
Receiving of first bit of data sent by the anchor node
$[\varkappa, \text{time}_{\text{proc}_u}]_{K_{a_j+}}$, Cert$_{u_i}$
time$_j$
time$_{\text{proc}_u}$
time$_{\text{proc}_a}$
Receiving of last bit and transmitting of first bit of the response
End of timer

Figure 27. Propagation time estimation process

4.2.3 TPM Based Geo-location

This algorithm is proposed by Sungjin Parka, Jong-Jin Wona, JaenamYoona, Kyong HoonKimb and Taisook Hanc [34]. The basic concept of this algorithm is to state the major problem of cloud services that is that the actual geolocation of cloud tenant devices can be easily manipulated. In general, an application requests the geolocation of a device to a GPS device driver.

In this process, there are many vulnerable points to forge the current geolocation of the device, which implies that the trusted computing base (TCB) for the trusted geolocation is too large. (e.g., the GPS device driver, system call tables, libraries for device driver communication, etc.) Since a large TCB-based system has high probability of embedding bugs, a secure system should minimize the TCB. The tiny hypervisor directly obtains the current geolocation from a GPS and computes an evidence value for the trusted geolocation with the Trusted Platform Module (TPM).

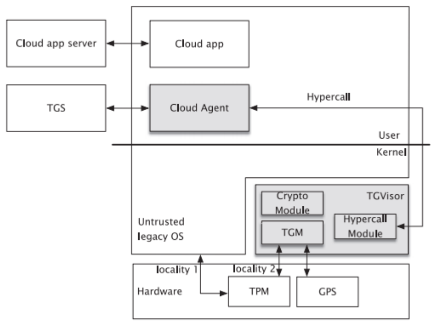


Figure 28. TPM Based Localization Framework

4.2.3.a Proposed Framework

Cloud tenants should install two proposed software, a tiny hypervisor, called *TGVisor*, and the Cloud Agent, in their devices. The hypervisor handles the geolocation value and performs TPM operations required for remote attestation. Locality 1 is assigned to the untrusted legacy OS and locality 2 to TGVisor. Throughout this locality assignment, the mobile device’s TPM can be shared by the tiny hypervisor and the untrusted legacy OS. TGVisor also includes the Crypto Module, a software cryptographic library that computes the evidence value for the trusted geolocation and creates a RSA session key. The Cloud Agent serves as a middleware to communicate between the hypervisor and the Trusted Geolocation Server (TGS), which is a verifier to check the trustworthiness of the TCB in target systems.

The TGS in the server-side periodically requests a trusted geolocation value and a remote attestation evidence to a cloud device. In turn, the Cloud Agent passes these requests to the hypervisor via hypercalls. The hypervisor obtains a geolocation value from the GPS connected to the tenant device and performs cryptographic operations inside the hypervisor. The hypervisor returns the results of the cryptographic operations to the Cloud Agent and the Cloud Agent transfers them to the TGS. The TGS attests the trustworthiness of the hypervisor and geolocation value with a public RSA session key and enforces a policy to the Policy DB running in a cloud provider domain. More information for this framework will be provided in chapter 6.

4.2.3.b Simulation Results

The trusted geolocation for cloud devices is a necessary feature to solve the security concerns of cloud users about the data location in the cloud. In order to cloud providers to provide more reliable data location services, TGVisor is presented as a novel trusted geolocation system for the cloud devices. TGVisor is feasible and practical in the cloud environment. Compared with other hypervisors based on XMHF, TGVisor was implement with the small LOC, 2293 LOC, which means that TGVisor maintains the minimized TCB.

4.2.4 Authenticated Location based on DRM

The idea of authenticated positioning and location utilizing Digital Rights Management (DRM) concept was proposed by Thomas Mundt [35], after finding a vast variety of scenarios where location is essential for controlling access to resources for example: A hard disc containing the blueprint of a nuclear bomb can only be read on the premises of the lab or TV shows or DVD movies are licensed to a single country only. This trusted position information is being used to enable access to data or devices protected by Digital Rights Management (DRM).

4.2.4.a Digital Rights Management (DRM)

Digital rights management (DRM) is a set of access control technologies for restricting the use of proprietary hardware and copyrighted works. Digital Rights Management allows the copyright owners of multimedia content to decide under which circumstances they want to allow users to access documents. Access can be restricted to read, write, change, update, and other operations. Managed material is secured by cryptographic methods such as encryption, watermarking, and signing. Encryption can be done by several algorithms such as RC4 and AES depending on the nature of the digital material. In order to determine an authenticated position, the following tasks must be accomplished in the given order:

1) A public key infrastructure (PKI) needs to be established.
2) The position needs to be calculated.
3) The position needs to be authenticated.
4) The DRM module decides whether it grants access to the protected material.

Creating a public key infrastructure: Each node carries a unique private key which will be used in the authentication process as well as for decryption of secret messages. The corresponding public key is signed by a Certifying Authority (CA) which belongs to each closed user group in our system. Traditional certificates such as X.509 can be used for this purpose.

Position determination: Signal strengths are utilized for positioning. Deriving the position directly from signal strengths of surrounding APs does not deliver accurate position information. An adapted method is utilized where uses a propagation model which is normally being calibrated by several test measurements. This calibrated model will be used to find the most likely position according to the current measurement of signal strengths. In order to calibrate the propagation model measurement reports are considered from nodes with known positions.

These nodes determine the signal strengths to other fixed nodes. The difference between expected signal strength and real signal strength is used to parameterize the propagation model. The error between expected and measured attenuation is virtually distributed over the entire distance between two nodes. By performing this within all nodes in sight of each node a two-dimensional model will be generated.

Authenticated positioning*:* As mentioned before all nodes are able to proof their identity by using a unique certificate which is signed by a CA. In order to proof its own position a node has to collect several measurement reports from surrounding nodes. Each measurement report contains the signal strength of the observed node as it is seen by the node generating the report. All reports are digitally signed using the node's private key. Some special nodes called ”level-0-nodes” have a certificate available that marks them as nodes with a position that is not doubtful.

Their position might have been securely determined by other means such as GPS or land surveying. The signatures of a CA ensures that only distinguished nodes can claim to be a ”level-0-node”. Nodes which derive their position from ”level-0-nodes” receive measurement reports as well as position reports. Both information are signed and therefore being marked as originated by a ”level-0-node”. All reports from nodes other than ”level-0-nodes” contain their calculated position (position report) and the signal strength of the node to which the report is addressed (measurement report). All reports are signed as usual by the sender. In order to proof the calculated position the sending node also includes the signed reports which were used to determine its own position. Following this scheme ensures that every node is able to see and check the paths which were used to determine its own position back to at least three level-0-nodes".

4.2.4.b Simulation Results

A prototype on a Linux driven TV satellite receiver with a built-in hard disc was implemented. The position was determinated and authenticated using a wireless mesh network. We are supporting a wireless community network with currently about 160 nodes (130 of them are stationary, the remaining are limited mobile. Commercial wireless routers running on Linux were used as nodes. The results were considered to be sufficient for the purpose of mesh network based location aware dependent digital rights management.

4.2.5 TOA-ECC Elliptic Curve Cryptography

Is a public key cryptography scheme for secure localization and authentication between sensor nodes proposed from V. Vijayalakshmi and Dr. T.G. Palanivelu [36]. The key exchange between the nodes is done by using ECC key Exchange. A comparison of this technique is also done with the other asymmetric algorithms like RSA and MPRSA. The exchange of the key is also done using Diffie-Hellman and then compared to prove that ECC is the best.

4.2.5.a Algorithm Overview

The primary reason for the attractiveness of ECC over systems such as RSA and DSA is that the best algorithm known for solving the underlying mathematical problem (namely, the ECDLP) takes fully exponential time. In contrast, sub exponential-time algorithms are known for underlying mathematical problems on which RSA and DSA are based, namely the integer factorization (IFP) and the discrete logarithm (DLP) problems.

This means that the algorithms for solving the ECDLP become infeasible much more rapidly as the problem size increases than those algorithms for the IFP and DLP. For this reason, ECC offers security equivalent to RSA and DSA while using far smaller key sizes. The attractiveness of ECC will increase relative to other public-key cryptosystems as computing power improvements force a general increase in the key size. The benefits of this higher-strength per-bit include:

- Higher speeds and Lower power consumption
- Bandwidth savings
- Storage efficiencies and Smaller Certificates

4.2.5.b Elliptic Curve Encryption and Decryption

To encrypt and send a message Pm to B, A chooses a random positive integer k and produces the cipher text Cm as given by equation consisting of the pair of points.

$$C_m = [kG, P_m + kP_B] \ (1)$$

Note that A has used B’s public key P_B. To decrypt the cipher text, B multiples the first point in the pair by B’s private key n_B and subtracts the result from the second point as shown by equation:

$$P_m + kP_B - n_B (kG) = P_m + k (n_B G) - n_B (kG) = P_m \ (2)$$

4.2.5.c ECC key exchange

A key exchange between users A and B can be accomplished as follows:

1. A selects an integer n_A less than n. This is A’s private key. A then generates a public key $P_A=n_A*G$; the public key is a point in Eq(a,b).
2. B similarly selects a private key nB and computes a public key PB,
3. The public keys are exchanged between the nodes A and B. A generates the secret key $K=n_A*P_B$. B generates the secret key $K=n_B*P_A$.

4.2.5.d Simulation Results

The TOA localization scheme along with ECC for secure localization and authentication was implemented. This TOA-ECC scheme was compared with the other public key cryptographic schemes like RSA and MPRSA. A further comparison was done by implementing both Diffie-Hellmann key exchange and ECC key exchange. The simulation results clearly indicate that TOA approach of localization along with the implementation of ECC with ECC key exchange is well suited for Wireless Sensor Networks.

4.2.6 Collaborative localization based on Trust model

This algorithm is proposed by Guangjie Han, Li Liu 1, Jinfang Jiang, Lei Shu, and Joel J.P.C. Rodrigues. CSLT [37] was implemented for Underwater Wireless Sensor Networks (UWSNs). First uses trust model to ensure node safety and avoid the influence from malicious nodes, which ultimately reduces unknown nodes’ localization error and enhances localization accuracy. Then, based on the collaboration of sensor nodes, localization ratio and localization accuracy can be further improved. The proposed CSLT consists of the following five sub-processes: trust evaluation of anchor nodes, initial localization of unknown nodes, trust evaluation of reference nodes, selection of reference node, and secondary localization of unknown node.

4.2.6.a Algorithm Overview

1. **Trust evaluation of anchor nodes**: In the first sub-process, the trust values of anchor nodes are calculated based on the main idea of detecting malicious anchor beacons.
2. **Initial localization of unknown nodes**: In the second sub-process, the unknown nodes are localized based on the multilateral localization method by using positioning reference information from trusty anchor nodes.
3. **Trust evaluation of reference nodes**: In the third sub-process, is evaluated the trust value for each successfully localized unknown node.

4. **Selection of reference node**: Then, the trusty and successfully localized unknown node can be selected as a reference node in the fourth sub-process.
5. **Secondary localization of unknown node**: Finally, in the fifth sub-process, two-hop trusty anchor nodes and reference nodes are used to help localize unknown nodes.

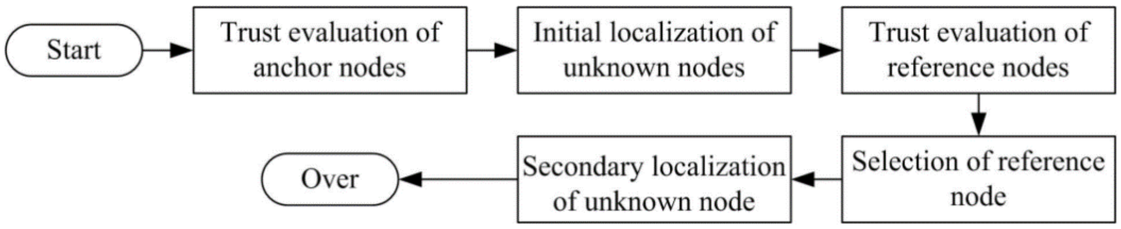

Start
Trust evaluation of anchor nodes
Initial localization of unknown nodes
Trust evaluation of reference nodes
Selection of reference node
Secondary localization of unknown node
Over

Figure 29. CSLT five sub-processes

4.2.6.b Simulation Results

The algorithm was implemented using MATLAB. In the experiments, the deployment area was set to 500m×500m×500m. There are 500 unknown nodes randomly deployed in the 3D space. The communication range of unknown nodes is set to 100 m. The performance of CSLT is compared based on the following three metrics:

(1) detect ratio of malicious nodes, (2) localization accuracy, (3) localization ratio, (4) energy consumption. Simulation results indicate that CSLT can achieve a high detect ratio of malicious nodes. In addition, the localization security including localization accuracy and localization ratio is improved in UWSNs. However, there are many remaining issues that need to be further studied.

4.2.7 Secure DV-Hop Localization algorithm

This algorithm is proposed by Xiaole Liu1, Rui Yang2 and Qingmin Cui [38]. Its basic idea realizes on transforming the distance to all beacon nodes from hops to meters by using computer average size of a hop. The advantages of the DV-Hop scheme are that it does not need any sophisticated hardware for the distance measurement and thus, it is free from range measurement errors. However, the DV-Hop technique introduces errors that propagated to the computation of a node’s location.

4.2.7.a Algorithm Overview

The proposed scheme includes four phases.

1. ***Initialization Phase***: Before the sensor nodes deployed, the sink node generates random keys for each beacon sensor node and applies the hash function to generate hash chain. The base station will store the hash function and the last key Rn of each key chain into all sensor nodes as the authentication key of the beacon node.
2. ***Hop-count Computation***: The goal of this phase is that all sensor nodes get the minimal hop-count to each beacon node. In this phase all beacon nodes broadcast its information to its neighbor nodes. Let sensor node B is a beacon node. All sensor node i will broadcast a message to its neighbors
3. ***Hop-size and Weighted Computation***: In this phase, all beacon nodes will compute the distance to the beacon nodes and the weight of beacon nodes.
4. ***Location Estimation***: An unknown sensor node can calculate its location when it has to estimate distance to at least three beacons and weights of the three beacons. The position of unknown nodes is computed using weighted least square method,

4.2.6.b Simulation Results

DV-Hop was compared with the proposed algorithm. The performance evaluation of the localization algorithm adopts an average positioning error as an evaluation index, as following formula:

$$e_i = \sum_{i=N}^{n} \frac{\sqrt{(x_i - \bar{x}_i)^2 + (y_i - \bar{y}_i)^2}}{r}$$

where *n* is the total number of sensor nodes in WSNs, *N* is the number of beacon nodes, (x_i, y_i) is the real coordinate of the unknown node *i* is, (x_i, y_i) is the evaluated coordinate, and *r* is the communication range of sensor nodes.

All the sensor nodes were random placed in a square area with the fixed size of 100*m*×100*m.* The radio range was 15 meters. And there are 20% beacon. The results demonstrate that the proposed algorithm can improve localization accuracy and against the nodes capture attacks effectively.

4.4 Conclusion

In this chapter we presented a variety of mechanisms where can provide safety and data protection to a geolocation system. Below we present a summary of which attacks, that were described in chapter 4.1, can be protected from the secure localization algorithms that were described above.

Table 5. Secure Localization algorithms VS Localization attacks

	Secure Localization Algorithms						
Attack Name	**LDEA**	**MAINV**	**TPMBG**	**ALB-DRM**	**TOA-ECC**	**CSLT**	**S. DV-HOP**
Compromise	✓	✓	✓	-	-	-	✓
Replication	-	-	-	-	-	✓	-
Impersonation	✓	✓	✓	✓	✓	✓	✓
Sybil attack	✓	✓	✓	✓	✓	✓	✓
Wormhole attack	✓	-	-	-	-	-	-
Forgery	✓	✓	✓	✓	-	✓	-
Alteration	✓	-	✓	-	✓	-	✓
Interference	-	-	-	-	-	-	-
Replay	✓	-	✓	-	✓	-	✓
Flooding	✓	-	✓	-	✓	-	✓
Selective forwarding	✓	-	✓	-	✓	-	✓
Stealing	✓	✓	✓	-	✓	✓	✓
Jamming	-	-	-	-	-	-	-
Collision	✓	-	✓	-	✓	-	✓
Exhaustion	-	✓	✓	-	-	-	-
Unfairness	✓	✓	✓	✓	✓	✓	✓
Dos Attacks	✓	-	✓	-	-	-	✓
Sinkhole	✓	✓	✓	✓	✓	✓	✓
Tampering	✓	✓	✓	-	-	-	✓
Insider attack	✓	✓	✓	✓	✓	✓	✓
Range change attack	✓	-	✓	-	-	-	-
False beacon location attack	✓	✓	✓	✓	✓	✓	✓
False reported location attack	✓	✓	✓	✓	✓	✓	✓
Desynchronization	✓	-	✓	-	✓	-	✓

The coordinates of the unknown node, certificates, TPM, public keys are some of the main ingredients for geolocation systems in order to achieve safety and security. In our proposed system, as it will be described in the next sessions, we decided to follow an approach utilizing the TPM concept for a safe geolocation system since it can provide a variety of ways for cryptography and safety algorithms alongside with key generations and strong encryptions methods. Summary is presented below:

Table 6. Localization frameworks summary.

Framework Name	Publication	Safety Mechanisms	Evaluation
Location-Dependent data Encryption Algorithm (LDEA)	**2008 [32]**	1. Toleration Distance (TD) 2. Latitude/Longitude 3. LDEA-key 4. Random keys	**Pros:** The secure provided key is truly random. Easy implementation. **Cons:** Is not strong enough as it uses the static location of mobile node and they are using the static tolerance distance to overcome the inaccuracy and inconsistent of GPS receiver.
Mutual Authentication and Insider Node Validation	**2017 [33]**	1. Node Authentication 2. Certificates	**Pros:** High efficiency in location estimation accuracy. Low relative error percentage with the increasing number of anchor nodes. Detections of malicious attack over 90%. supports mobility of the nodes and therefore it is suitable for dynamic network environments. **Cons:** High complexity and computation overhead
Trust Platform Module (TPM) Based Geo-location	**2016 [34]**	1. Trust Platform Module 2. Cryptography 3. Cloud Agent	**Pros:** Is feasible and practical. Provides strong cryptographical mechanisms and algorithms and has high performance and low computations errors. **Cons:** Is not suitable for indoor positioning since it loses significant power inside buildings due to GPS and has computation overhead due to TPM operations.
Authenticated Location based on DRM	**2005 [35]**	1. Public key 2. Node Authentication 3. Digital Rights Management (DRM)	**Pros:** Very effective for mesh networks. High performance for position authentication by a web trust protocol DRM. **Cons:** No measure system to describe expected and demanded confidence of the indicated position. No upper limit for the position error depends from the geographical configuration
TOA-ECC Elliptic Curve Cryptography	**2008 [36]**	1. Public key 2. Elliptic Cryptography	**Pros:** High performance, very low encryption and decryption time. Easy implementation. Is well suited for Wireless Sensor Networks. **Cons:** Requires highly accurate synchronization of sender and receiver clocks due to TOA. High possibility of computations errors.
Collaborative Secure Localization algorithm based on Trust model (CSLT)	**2016 [37]**	1. Trust model 2. Trusty anchors nodes 3. References node	**Pros:** High localization accuracy and localization ratio in UWSNs. High detect ratio of malicious nodes. **Cons:** Cannot find each type of malicious nodes with 100%. Not very efficient if many malicious nodes launch an attack simultaneously. Tested only in MATLAB not to other platforms.
Secure DV-Hop Localization for WSN	**2015 [38]**	1. DV-Hop concept 2. Random keys	**Pros:** High effectiveness under different attack and high performance of localization. High localization accuracy and strong against localization attacks. **Cons:** Applicable only to DV-Hop localization process. Not scalable to other frameworks.

5. HYPERVISORS AND GEOLOCATION

In this chapter we present a short overview on Hypervisors and how they achieved security and safety in systems. We present the types of Hypervisor alongside with information of some representative cases.

5.1 Hypervisors Overview

Hypervisor, also known as a virtual machine monitor, is a process that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, like memory and processing. Generally, there are two types of hypervisors. Type 1 hypervisors, called “bare metal,” run directly on the host’s hardware.

Type 2 hypervisors, called “hosted,” run as a software layer on an operating system, like other computer programs. Hypervisors make it possible to use more of a system’s available resources, and provide greater IT mobility, since the guest VMs are independent of the host hardware. This means they can be easily moved between different servers.

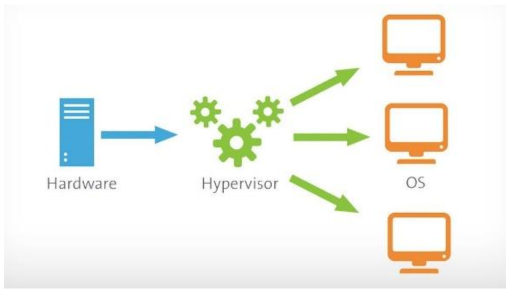


Figure 30. Hypervisor Overview

Hypervisors provide several benefits to the enterprise data center. First, the ability of a physical host system to run multiple guest VMs can vastly improve the utilization of the underlying hardware. Where physical (nonvirtualized) servers might only host one operating system and application, a hypervisor virtualizes the server, allowing the system to host multiple VM instances -- each running an independent operating system and application -- on the same physical system using far more of the system's available compute resources. VMs are also very mobile. The abstraction that takes place in a hypervisor also makes the VM independent of the underlying hardware. Traditional software can be tightly coupled to the underlying server hardware, meaning that moving the application to another server requires time-consuming and error-prone reinstallation and reconfiguration of the application.

By comparison, a hypervisor makes the underlying hardware details irrelevant to the VMs. This allows any VMs to be moved or migrated between any local or remote virtualized servers -- with sufficient computing resources available -- almost at-will with effectively zero disruption to the VM; a feature often termed live migration. VMs are also logically isolated from each other -- even though they run on the same physical machine. In effect, a VM has no native knowledge or dependence on any other VMs. An error, crash or malware attack on one VM does not proliferate to other VMs on the same or other machines. This makes hypervisor technology extremely secure. Finally, VMs are easier to protect than traditional applications.

A physical application typically needs to be first quiesced and then backed up using a time-consuming process that results in substantial downtime for the application. A VM is essentially little more than code operating in a server's memory space. Snapshot tools can quickly capture the content of that VM's memory space and save it to disk in moments -- usually without quiescing the application at all. Each snapshot captures a point-in-time image of the VM which can be quickly recalled to restore the VM on demand.

Hypervisors are traditionally implemented as a software layer, but hypervisors can also be implemented as code embedded in a system's firmware. There are two principal types of hypervisor. Type 1 hypervisors are deployed directly atop the system's hardware without any underlying operating systems or other software. These are called "bare metal" hypervisors and are the most common and popular type of hypervisor for the enterprise data center. Examples include vSphere or Hyper-V. The first hypervisors, which IBM developed in the 1960s, were native hypervisors.

These included the test software SIMMON and the CP/CMS operating system (the predecessor of IBM's z/VM). Modern equivalents include AntsleOs, Xen, XCP-ng, Oracle VM Server for SPARC, Oracle VM Server for x86, Microsoft Hyper-V, Xbox One system software, and VMware ESX/ESXi. Type 2 hypervisors run as a software layer atop a host operating system and are usually called "hosted" hypervisors like VMware Player or Parallels Desktop. Hosted hypervisors are often found on endpoints like PCs. VMware Workstation, VMware Player, VirtualBox, Parallels Desktop for Mac and QEMU are examples of type-2 hypervisors.

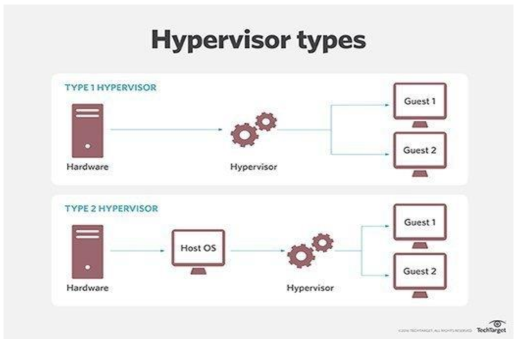


Figure 31. Hypervisor types

Hypervisors are important to any system administrator or system operator because virtualization adds a crucial layer of management and control over the data center and enterprise environment. Staff members not only need to understand how the respective hypervisor works, but also how to operate supporting functionality such as VM configuration, migration and snapshots. The role of a hypervisor is also expanding. For example, storage hypervisors are used to virtualize all of the storage resources in the environment to create centralized storage pools that administrators can provision -- without having to concern themselves with where the storage was physically located.

Today, storage hypervisors are a key element of software-defined storage. Networks are also being virtualized with hypervisors, allowing networks and network devices to be created, changed, managed and destroyed entirely through software without ever touching physical network devices. As with storage, network virtualization is appearing in broader software-defined network or software-defined data center platforms.

5.2 Trust Platform Module

In order to the hypervisor to achieve all the security and cryptographical processes can utilize the usage of a Trust Platform Module (TPM). The TPM is a crypto processor designed to secure hardware and for the creation and generation of cryptographic keys. Here we present a short brief about what a TPM is.

5.2.1 Introduction on Trust Platform Module

Trusted Platform Module (TPM, also known as ISO/IEC 11889) is an international standard for a secure crypto processor, a dedicated microcontroller designed to secure hardware through integrated cryptographic keys. Trusted Platform Module (TPM) was conceived by a computer industry consortium called Trusted Computing Group (TCG), and was standardized by International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) in 2009 as ISO/IEC 11889. [39]

TCG continued to revise the TPM specifications. The last revised edition of TPM Main Specification Version 1.2 was published on March 3, 2011. It consisted of three parts, based on their purpose.[2] For the second major version of TPM, however, TCG released TPM Library Specification 2.0, which builds upon the previously published TPM Main Specification. Its latest edition was released on September 29, 2016, with several errata with the latest one being dated on January 8, 2018.

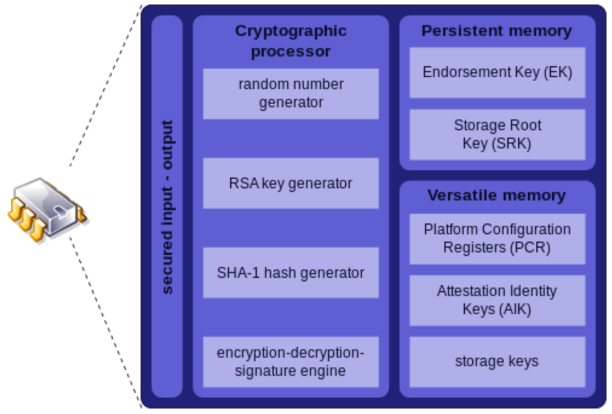


Figure 32. Trust Platform Module overview

Trusted Platform Module provides

- A random number generator
- Facilities for the secure generation of cryptographic keys for limited uses.
- Remote attestation: Creates a nearly unforgeable hash key summary of the hardware and software configuration. The software in charge of hashing the configuration data determines the extent of the summary. This allows a third party to verify that the software has not been changed.
- Binding: Encrypts data using the TPM bind key, a unique RSA key descended from a storage key.
- Sealing: Similar to binding, but in addition, specifies the TPM state for the data to be decrypted (unsealed).

Computer programs can use a TPM to authenticate hardware devices, since each TPM chip has a unique and secret RSA key burned in as it is produced. Pushing the security down to the hardware level provides more protection than a software-only solution.

5.2.2 TPM implementations

The United States Department of Defense (DoD) specifies that "new computer assets (e.g., server, desktop, laptop, thin client, tablet, smartphone, personal digital assistant, mobile phone) procured to support DoD will include a TPM version 1.2 or higher where required by DISA STIGs and where such technology is available." DoD anticipates that TPM is to be used for device identification, authentication, encryption, and device integrity verification.

5.2.2.a Platform integrity

The primary scope of TPM is to assure the integrity of a platform. In this context, "integrity" means "behave as intended", and a "platform" is any computer device regardless of its operating system. It is to ensure that the boot process starts from a trusted combination of hardware and software, and continues until the operating system has fully booted and applications are running.

The responsibility of assuring said integrity using TPM is with the firmware and the operating system. For example, Unified Extensible Firmware Interface (UEFI) can use TPM to form a root of trust: The TPM contains several Platform Configuration Registers (PCRs) that allow secure storage and reporting of security relevant metrics. These metrics can be used to detect changes to previous configurations and decide how to proceed. Good examples can be found in Linux Unified Key Setup (LUKS), BitLocker and Private Core vCage memory encryption.

An example of TPM use for platform integrity is the Trusted Execution Technology (TXT), which creates a chain of trust. It could remotely attest that a computer is using the specified hardware and software.

5.2.2.b Disk encryption

Full disk encryption utilities, such as dm-crypt and BitLocker, can use this technology to protect the keys used to encrypt the computer's storage devices and provide integrity authentication for a trusted boot pathway that includes firmware and boot sector.

5.2.2.c Password protection

Operating systems often require authentication (involving a password or other means) to protect keys, data or systems. If the authentication mechanism is implemented in software only, the access is prone to dictionary attacks. Since TPM is implemented in a dedicated hardware module, a dictionary attack prevention mechanism was built in, which effectively protects against guessing or automated dictionary attacks, while still allowing the user a sufficient and reasonable number of tries. Without this level of protection, only passwords with high complexity would provide sufficient protection.

5.2.2.d Other uses and concerns

Any application can use a TPM chip for:

- Digital rights management
- Protection and enforcement of software licenses
- Prevention of cheating in online games[13]

Other uses exist, some of which give rise to privacy concerns. The "physical presence" feature of TPM addresses some of these concerns by requiring BIOS-level confirmation for operations such as activating, deactivating, clearing or changing ownership of TPM by someone who is physically present at the console of the machine

Starting in 2006, many new laptops have been sold with a built-in TPM chip. In the future, this concept could be co-located on an existing motherboard chip in computers, or any other device where the TPM facilities could be employed, such as a cellphone. On a PC, either the LPC bus or the SPI bus is used to connect to the TPM chip. TCG has certified TPM chips manufactured by Infineon technologies, Nuvoton, and STMicroelectronics, having assigned TPM vendor IDs to Advanced micro devices ,Atmel, Intel ,Broadcom , IBM, Infineon, Lenovo, National Semiconductor, Nationz Technologies, Nuvoton ,Qualcomm, Rockchip, Standard Microsystems Corporation, STMicroelectronics, Samsung, Sinosun, Texas Instruments, and Winbond.

There are five different types of TPM 2.0 implementations:

- **Discrete TPMs** are dedicated chips that implement TPM functionality in their own tamper resistant semiconductor package. They are theoretically the most secure type of TPM because the routines implemented in hardware should be more resistant to bugs versus routines implemented in software, and their packages are required to implement some tamper resistance.
- **Integrated TPMs** are part of another chip. While they use hardware that resists software bugs, they are not required to implement tamper resistance. Intel has integrated TPMs in some of its chipsets.

- **Firmware TPMs** are software-only solutions that run in a CPU's trusted execution environment. Since these TPMs are entirely software solutions that run in trusted execution environments, these TPMs are more likely to be vulnerable to software bugs. AMD, Intel and Qualcomm have implemented firmware TPMs.
- **Software TPMs** are software emulators of TPMs that run with no more protection than a regular program gets within an operating system. They depend entirely on the environment that they run in, so they provide no more security than what can be provided by the normal execution environment, and they are vulnerable to their own software bugs and attacks that are penetrating the normal execution environment. They are useful for development purposes.
- **Virtual TPMs** are provided by a hypervisor. Therefore, they rely on the hypervisor to provide them with an isolated execution environment that is hidden from the software running inside virtual machines to secure their code from the software in the virtual machines. They can provide a security level comparable to a firmware TPM.

5.3 Trusted Hypervisors

5.3.1 XMHF- uberXMHF

5.3.1.a XMHF

XMHF is an eXtensible and Modular Hypervisor Framework that strives to be a comprehensible and flexible platform for performing hypervisor research and development. The framework allows others to build custom (security-sensitive) hypervisor-based solutions (called "hypapps"). The XMHF is capable of running unmodified legacy multiprocessor capable OSes such as Windows and Linux. The XMHF core has a TCB of 6018 SLoC, and its performance is comparable. [40]

XMHF is designed to achieve three goals – modular extensibility, automated verification, and high performance. XMHF includes a core that provides functionality common to many hypervisor-based security architectures and supports extensions that augment the core with additional security or functional properties while preserving the fundamental hypervisor security property of memory integrity (i.e., ensuring that the hypervisor’s memory is not modified by software running at a lower privilege level).

XMHF advocates a "rich" single-guest execution model where the hypervisor framework supports only a single-guest and allows the guest direct access to all performance-critical system devices and device interrupts. XMHF currently runs on recent multicore x86 hardware virtualized platforms with support for dynamic root of trust and nested (2-dimensional) paging.

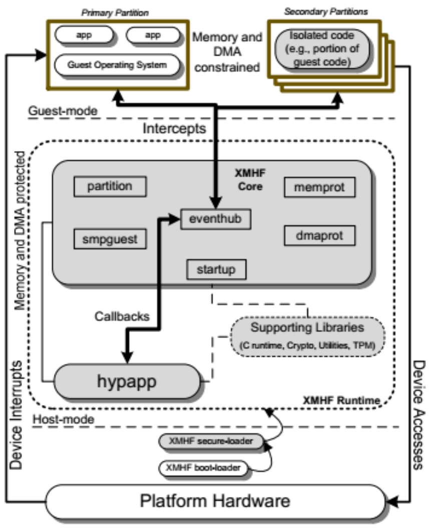


Figure 33. XMHF platform architecture

5.2.2.b XMHF Framework Overview

XMHF consists of the XMHF core and small supporting libraries that sit directly on top of the platform hardware. A hypapp extends the XMHF core to implement the desired (security) functionality. XMHF allows the guest direct access to all performance-critical system devices and device interrupts resulting in reduced hypervisor complexity, consequently Trusted Computing Base (TCB), as well as high guest performance. The high-level design principles behind XMHF are platform independent. The XMHF implementation currently supports both Intel and AMD x86 hardware virtualized platforms, and unmodified multi-processor Windows (2003 and XP) and Linux as guests. However, XMHF design principles apply to other architectures, such as ARM, as well.

5.2.2.c Hypervisor Properties Required by DRIVE

DRIVE (Designing hypervisors for Rigorous Integrity VErification) is composed of a set of hypervisor properties and system invariants. The hypervisor properties entail the invariants, which in turn imply the hypervisor’s memory integrity.

The virtualized system is modeled as a tuple V = (H, G, D, M), where H is the hypervisor, G represents the guest, D represents devices, and M is the hypervisor memory containing both hypervisor code and data. Both G and D are controlled by the attacker. The guest memory is separate from M and irrelevant to memory integrity, from the model. DRIVE consists of a set of properties about H, system invariants, and a proof that if H satisfies those properties then the invariants hold on all executions of V. This, in turn, implies the memory integrity of H in V. DRIVE identifies the following six properties that restrict the hypervisor design and implementation:

1. **Modularity (MOD**). Upon hypervisor initialization, control is transferred to a function init(). When an intercept is triggered, the hardware transfers control to one of the intercept handlers ih1(), . . .ihk().

2. **Atomicity (ATOM).** This property ensures the atomicity of initialization and intercept handling on the CPU(s). It consists of two sub-properties: $ATOM_{init}$ – at the start of V 's execution, init( ) runs completely in a single-threaded environment before any other code executes; $ATOM_{ih}$ – the intercept handlers ih1(), . . . , ihk() always execute in a single-threaded environment.
3. **Memory Access Control Protection (MPROT).** H uses a memory access control mechanism MacM. All MacM related state is stored in M. MacM consists of two parts:
    a. MacMG – for the guest
    b. MacMG – for the devices.
4. **Correct Initialization (INIT).** After H's initialization, MacM protects M from the guest and devices. The intercept entry points into H points to the correct intercept handler.
5. **Proper Mediation (MED).** MacM is active whenever attacker-controlled programs execute. This implies:
    a. Before control is transfered to the guest (G), the CPU is set to execute in guest mode to ensure that MacMG is active,
    b. MacMD is always active.
6. **Safe State Updates (SAFEUPD).** All updates to system state including M and control structures of the hardware TCB (e.g., guest execution state and chipset I/O), by an intercept handler:
    a. preserve the protection of M by MacM in guest mode and for all devices;
    b. do not modify the intercept entry point into H,
    c. do not modify H's code.

The design and implementation decisions that help make XMHF minimalistic, enable verification of DRIVE properties on XMHF's C implementation, and make automated re-verification in the process of hypapp development possible. XMHF is a Type-1 (or native, bare metal) hypervisor that runs directly on the host's hardware to control the hardware and to manage a guest OS. The guest runs on another (unprivileged) level above the hypervisor.

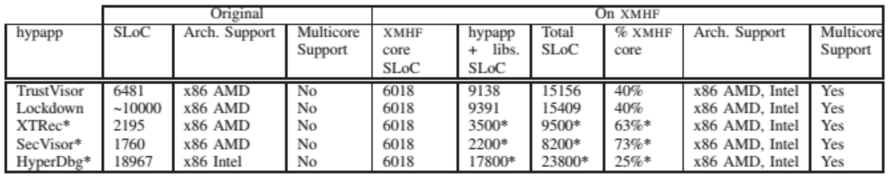
The baremetal design allows for a small-TCB and high performance hypervisor code base. Recall that XMHF consists of the XMHF core and small supporting libraries that sit directly on top of the platform hardware. A hypapp extends the XMHF core and leverages the basic hypervisor and platform functionality provided by the core to implement the desired (security) functionality OSes.

To achieve DRIVE properties, XMHF relies on platform hardware support, which includes hardware virtualization, two-level Hardware Page Tables (HPT), DMA protection, and dynamic root of trust (DRT) support. These capabilities are found on recent Intel and AMD x86 platforms. Similar capabilities are also forthcoming in ARM processor platforms. While this breaks backward compatibility with older hardware, it allows XMHF's design to be much smaller and cleaner while achieving the DRIVE properties to ensure memory integrity.

5.2.2.d XMHF Framework Evaluation

XMHF’s TCB consists of the XMHF core, the hypapp and supporting libraries used by the hypapp. The XMHF supporting libraries (totaling around 8K lines of C code) currently include a tiny C runtime library, a small library of cryptographic functions, a library with optional utility functions such as hardware page table abstractions and command line parsing functions, and a small library to perform useful TPM operations.

From a hypapp’s perspective, the minimum TCB exposed by XMHF comprises the XMHF core which consists of 6018 SLoC. The figure below shows that the XMHF core forms 48% of a hypapp’s TCB, on average. This supports the hypothesis that these hypervisors share a common hypervisor core that is re-used or engineered from scratch with every new application.



	Original			On XMHF					
hypapp	SLoC	Arch. Support	Multicore Support	XMHF core SLoC	hypapp + libs. SLoC	Total SLoC	% XMHF core	Arch. Support	Multicore Support
TrustVisor	6481	x86 AMD	No	6018	9138	15156	40%	x86 AMD, Intel	Yes
Lockdown	~10000	x86 AMD	No	6018	9391	15409	40%	x86 AMD, Intel	Yes
XTRec*	2195	x86 AMD	No	6018	3500*	9500*	63%*	x86 AMD, Intel	Yes
SecVisor*	1760	x86 AMD	No	6018	2200*	8200*	73%*	x86 AMD, Intel	Yes
HyperDbg*	18967	x86 Intel	No	6018	17800*	23800*	25%*	x86 AMD, Intel	Yes

Figure 34. Porting status of several HyperVisors.

It was measured XMHF’s runtime performance using two metrics:

1. guest overhead imposed solely by the framework (i.e., without any hypapp),
2. base overhead imposed by XMHF for a given hypapp.

The platform was an HP Elitebook 8540p with a Quad-Core Intel Core i7 running at 3 GHz, 4 GB RAM, 320GB SATAHDD and an Intel e1000 ethernet controller, using Ubuntu12.04 LTS as the guest OS running the Linux kernel v3.2.2.For network benchmarks, it was connected another machine via a1 Gbps Ethernet crossover link and run the 8540p as a server. It was used XMHF with both 4K and 2MB hardware page table (HPT) mappings for measurement purposes. Most of the SPEC benchmarks show less than 3% performance overhead. However, there are four benchmarks with over 10%, and two more with 20% and 55% overhead. For I/O application benchmarks, read access to files and network access incurs the highest overhead (40% and 25% respectively). The rest of the benchmarks show less than 10% overhead.

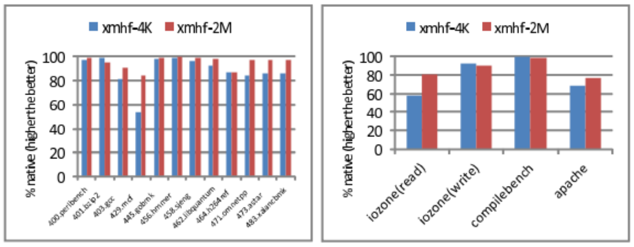


Figure 35. XMHF Application Benchmarks

XMHF's performance was compared with the popular Xen (v 4.1.2) hypervisor. Three hardware virtual machine (HVM) configurations were used for domU, that are identical in memory and CPU configuration to the native system: HVM domU (xen-domU-hvm), HVM domU with para virtualized drivers (xen-domU-pvhvm) and HVM domU with pci passthrough (xen-domU-passthru). dom0 was also used (xendom0) as a candidate for performance evaluation.

For compute-bound applications XMHF and Xen have similar overheads (around 10% on average) with the 2MB XMHF HPT configuration performing slightly better. For disk I/O benchmarks, XMHF, xen-dom0 and xendom U-pvhvm have the lowest overheads (ranging from 3-20%). Both XMHF and Xen have higher overheads on the disk read benchmark when compared to other disk benchmarks. For network I/O benchmark, XMHF has the lowest overhead (20-30%). xen-dom0 and xen-domU-passthru incur a 45% and 60% overhead respectively, while xen-domU-hvm and xen-domU-pvhvm have more than 85% overhead.

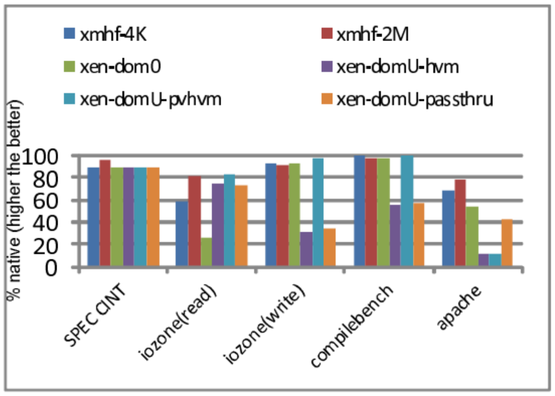


Figure 36. XMHF performance comparison with Xen

5.2.2.e uberXMHF

XMHF is no longer in active development. It is superseded by uberXMHF (uber eXtensible Micro-Hypervisor Framework). The uber eXtensible Micro-Hypervisor Framework (uberXMHF) is a compositionally verifiable, extensible, micro-hypervisor framework for commodity platforms advocating the design and development of a new class of security-oriented micro hypervisor base applications ("uberapps").

uberXMHF is designed to achieve three goals: modular extensibility, automated (compositional) verification, and high performance. uberXMHF includes a core that provides functionality common to many hypervisor-based security architectures and supports extensions that augment the core with additional security or functional properties while preserving the fundamental hypervisor security property of memory integrity (i.e., ensuring that the hypervisor's memory is not modified by software running at a lower privilege level).

uberXMHF advocates a “rich” commodity single-guest execution model (uber-guest) where the hypervisor framework supports only a single, commodity guest OS and allows the guest direct access to all performance-critical system devices and device interrupts. In principle, the uber-guest could also be a traditional hypervisor/VMM. uberXMHF currently runs on both x86 (Intel and AMD) and ARM (Raspberry PI) multi-core hardware virtualized platforms with support for nested (2-dimensional) paging. The framework can run unmodified legacy multiprocessor capable OSes such as Linux and Windows.

5.3.2 Xvisor

Xvisor is an open-source type-1 hypervisor, which aims at providing a monolithic, light-weight, portable, and flexible virtualization solution. It provides a high performance and low memory footprint virtualization solution for ARMv5, ARMv6, ARMv7a, ARMv7a-ve, ARMv8a, x86_64, and other CPU architectures. In comparison to other ARM hypervisors, it is one of the few hypervisors providing support for ARM CPUs which do not have ARM virtualization extensions [41].

The Xvisor source code is highly portable and can be easily ported to most general-purpose 32-bit or 64-bit architectures as long as they have a paged memory management unit (PMMU) and a port of the GNU C compiler (GCC). Xvisor primarily supports Full virtualization hence, supports a wide range of unmodified Guest operating systems. Paravirtualization is optional for Xvisor and will be supported in an architecture independent manner (such as VirtIO PCI/MMIO devices) to ensure no-change in Guest OS for using para virtualization.

It has most features expected from a modern hypervisor, such as: Device tree based configuration, and high resolution timekeeping, Threading framework, Host device driver framework, IO device emulation framework, Runtime loadable modules, Pass through hardware access, Dynamic guest creation/ destruction , Management terminal, Network virtualization, Input device virtualization, Display device virtualization and many more. Hypervisors can be categorized into three categories based on Host hardware access, CPU virtualization, and Guest IO emulation, as follows:

1. ***Complete Monolithic*:** Complete monolithic hypervisors (e.g. Xvisor) have one common software for Host hardware access, CPU virtualization, and Guest IO emulation.
2. ***Partially Monolithic*:** Partially monolithic hypervisors (e.g. KVM) are usually an extension of the general purpose of monolithic OS (e.g. Linux, FreeBSD, NetBSD, etc.) to support Host hardware access + CPU virtualization in kernel and support Guest IO emulation from software running in user-space (e.g. QEMU).
3. ***Micro-kernelized:*** Micro-kernelized hypervisors (e.g. Xen) are usually light-weight micro-kernels providing basic Host hardware access + CPU virtualization in kernel and for rest it depends on a Management Guest (e.g. Dom0 of Xen)

which provides complete Host hardware access, Management interface, and Guest IO emulation.

Xvisor is a complete monolithic hypervisor whereas most open-source hypervisors are either partially monolithic or micro-kernelized.

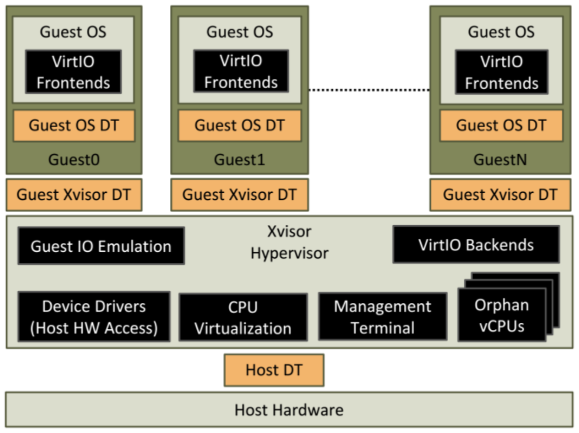


Figure 37. XVisor Architecture Overview

5.3.2.a Xvisor Framework Overview

All core components of Xvisor such as: CPU virtualization, guest IO emulation, background threads, para-virtualization services, management services, and device drivers run as a single software layer with no prerequisite tool or binary file. The guest OS runs on what Xvisor implementers call Normal vCPUs, having a privilege less than Xvisor. Moreover, all background processing for device drivers and management purposes run on Orphan vCPUs with highest privilege. Guest configuration is maintained in the form of a tree data structure called device tree [21]. This facilitates easier manipulation of guest hardware through device tree script (DTS). In other words, no source code changes are required for creating a customized guest for embedded systems.

The most important advantage of Xvisor is its single software layer running with highest privilege, in which all virtualization related services are provided. Unlike KVM, Xvisor's context switches are very lightweight (refer to section V) resulting in fast handling of nested page faults, special instruction traps, host interrupts, and guest IO events. Furthermore, all device drivers run directly as part of Xvisor with full privilege and without nested page table (unlike Xen) ensuring no degradation in device driver performance. In addition, the Xvisor vCPU scheduler is per-CPU and does not do load balancing for multiprocessor systems.

The multi-processor load balancer is a separate entity in Xvisor, independent of the vCPU scheduler (unlike KVM and Xen). Both, vCPU scheduler and load balancer are extensible in Xvisor. Xvisor's only limitation is its lack of rich board and device driver support like Linux. To tackle this limitation Xvisor provides Linux compatible headers for porting device driver frameworks and device drivers from Linux kernel. Albeit not completely solving the problem, porting efforts are greatly reduced.

5.3.2.b Host Interrupts

Xvisor's host device drivers generally run as part of Xvisor with highest privilege. Hence, no scheduling or context switch overhead is incurred for processing host interrupts. A scheduling overhead only incurs if the host interrupt is routed to guest, which is not running currently.

5.3.2.c Memory Management

Xvisor ARM pre-allocates contiguous host memory as guest RAM at guest creation time. It creates a separate three level stage2 translation table for each guest. Xvisor ARM can create 4KB or 2MB or 1GB translation table entries in stage2. Additionally, it always creates the biggest possible translation table entry in stage2 based on IPA and PA alignment. Finally, the guest RAM being flat/contiguous (unlike other hypervisors) helps cache speculative access, which further improves memory accesses for guests.

5.3.2.e Memory Footprint Comparison

Embedded systems require small memory footprint:

Xvisor ARM	**Installations**
	Xvisor kernel[c]
Size	1-2 MB
Memory on cubieboard2	4+ MB out of 16MB[d]

Table 7. XVisor Memory Footprint

5.3.2.f Xvisor Framework Evaluation

The experiments aimed to evaluate the newly proposed embedded hypervisor Xvisor's efficiency in comparison to KVM and Xen. Four benchmark applications were tried on guest Linux running on Cubieboard2 [25]. The Cubieboard2 is an ARM Cortex-A7 dual core 1GHz board with 1GB RAM. The following hypervisor versions are used in our experiments:

1. **KVM:** Latest Linux-3.16-rc3 is used as Host KVM kernel. The guest kernel is Linux-3.16-rc3.
2. **Xen:** Latest Xen-4.5-unstable kernel dated 3rd August 2014 is used as hypervisor. The Dom0 kernel is Linux-3.16-rc3 and DomU kernel is also Linux-3.16-rc3.
3. **Xvisor:** Latest Xvisor-0.2.4+ dated 18th July 2014 is used as hypervisor. The guest kernel is Linux-3.16-rc3.

Experimental results are obtained with two test vectors. The first runs over a single core, while the second runs over a dual core. The systems under test (SUTs) are:

1. Host without any hypervisor
2. Xvisor guest
3. KVM guest

4. KVM guest with HugeTLB
5. Xen guest.

In order to ensure that only CPU overhead, memory bandwidth and lock synchronization latency are taken into consideration, both test vectors have one para virtualized guest with two vCPUs. Moreover, all hypervisors have the following optimizations: No maintenance interrupt from generic interrupt controller, Super pages support for Xen ARM, and Trap-and-yield vCPU on WFE instruction.

The DMIPS obtained on Xvisor guest are around 0.2% higher than KVM guest, 0.19% higher than KVM guest with HugeTLB, and 0.46% higher than Xen DomU. The Dhrystone benchmark is small in size and mostly fits in cache at runtime hence memory access overhead does not affect it. Despite obtaining improvement of 2 DMIPS, this still improves the overall system performance because 1 DMIPS equals 1757 iterations-per-second.

Therefore, actual improvement will be thousands of Dhrystone iterations (typically few million machine cycles). The memory copy results of Xvisor guest are around 18% higher than KVM guest,1.2% higher than KVM guest with HugeTLB, and 0.67%higher than Xen DomU. Also, integer read-modify-writer esults of Xvisor guest are around 1.14% higher than KVMguest, 1.2% higher than KVM guest with HugeTLB, and1.64% higher than Xen DomU.

Results show sustainable memory bandwidth in Xvisor guest are around 0.72%higher than KVM guest, 1.57% higher than KVM guest with HugeTLB and 1.2% higher than Xen DomU. Hackbench results show that task dispatch latency on Xvisor guest is around 12.5% lower than KVM guest, 5.62% lower than KVMguest with HugeTLB and 6.39% lower than Xen DomU.

5.3.3 TGVisor

The major problem with trust geo-location between service providers and cloud tenants is that the actual geo-location of the cloud tenant device can be easily manipulated. In the process of geo-location, there are many vulnerable points to forge the current geo-location of the devices which means that the trusted computing base (TCB) for the trusted geo-location is too large. Since a large TCB-based system has high probability of embedding bugs, a secure system should minimize the TCB. [42]

TGVisor is a proposed framework from Sungjin Park, Jae Nam Yoon, Cheoloh Kang, Kyong Hoon Kimand Taisook Han[]. The main features of TGVisor are a hardware-assisted tiny hypervisor, the Dynamic Root of Trust Management (DRTM), and the TPM. With the combination of these components, TGVisor delivers the trusted geo-location of the mobile cloud devices to the cloud provider.

It uses the TPM-based remote attestation and the hypervisor-based trusted geo-location module in order to guarantee its trustworthiness. The key role of TGVisor is to compute the remote attestation value of the TCB and the current geo-location based on the TPM. The TGVisor handles the geo-location value and performs TPM operations required for the remote attestation. The primary goal of the remote attestation is to guarantee the trustworthiness of TGVisor and the integrity of the current geo-location value.

5.3.3.a Architecture

The tiny hypervisor assigns the locality 1 to the untrusted legacy OS and the locality 2 to the tiny hypervisor. Throughout this locality assignment, the mobile device's TPM can be shared by the tiny hypervisor and the untrusted legacy OS. The Cloud Agent serves as a middleware to communicate between the tiny hypervisor and the Trusted Geo-location Server, which is a verifier to check the trustworthiness of the TCB in the target system. The TGS in the server-side periodically requests a trusted geo-location value and a remote attestation result. In turn, the Cloud Agent passes these requests to the hypervisor via hypercalls.

The hypervisor obtains a geo-location value from the GPS connected to the tenant devices and performs cryptographic operations based on the TPM. The hypervisor returns the results of the cryptographic operations to the Cloud Agent which again transfers them to the TGS. The TGS attests to the trustworthiness of the hypervisor and the geo-location value and enforces a policy to the Policy DB running in the cloud provider domain.

In order to ensure the trusted geo-location, all the operations involved in the trusted geo-location must exist in or be isolated by the hypervisor. They must place in a separate VM including a secure OS, a TPM library, a TPM driver, and an attestation application. Since this makes the TCB, the separate VM, larger, it violates the premise that the minimized TCB is more reliable.

TGVisor includes two key modules to meet the above requirements: the Hypercall Module and the Trusted Geo-location Module. The Hypercall Module serves as a gateway for TGVisor to communicate with outer applications such as the Cloud Agent. It contains functions to process requests from the Cloud Agent, namely hyper_loadkey, hyper_quote2, and hyper_getgeoloc. The TGM is a set of functions to compute the trusted geo-location of the mobile device.

The key functions of the TGM can be divided into the secure geo-location reading and the cryptographic operation. The TGM retrieves the current geo-location of the mobile cloud device from the actual geo-location sensor like the GPS. Even though the TGM is totally protected by the hypervisor, a geo-location value derived from the TGM is not guarded during the transmission to a location server. For the secure transmission, the TGM performs a cryptographic sign operation with the AIK based on the TPM.

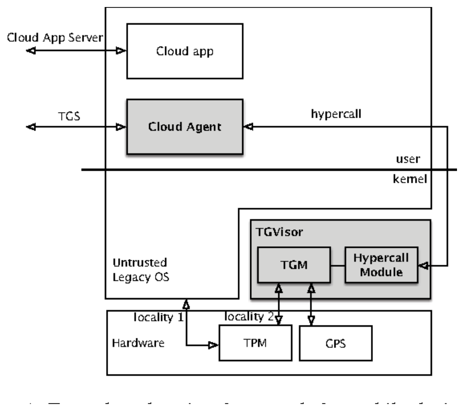


Figure 38. TGVisor architecture

5.3.3.b Attestation protocol

The proposed framework measures the trustworthiness of TGVisor using the integrity of TGVisor with the SKINIT instruction. Additionally, the TGS must validate that a geo-location value from the tenant device is intact. The hash function is the most general way to check the intactness, so TGVisor leverages the hash of the geo-location value [43]. The trusted geo-location is assured between the TGS and the mobile cloud client by the attestation protocol described by the following steps:

1. Preparing the attestation protocol where a mobile device launches TGVisor with the SKINIT instruction, at boot time as a result if attackers attempt to load their own hypervisors with the SKINIT, PCR17 must be modified and the TGS can detect this abnormal behavior with the attestation protocol for the trusted geo-location.
2. Establishing a secure session when a cloud user runs a cloud app, throughout this step, the TGS and the Cloud Agent mutually authenticate each other and share secrets to be used to protect the communication by Transport Layer Security (TLS). Even that the Cloud Agent and TGVisor path is not secure because the attestation value is cryptographically protected by AIK and TPM, TGVisor, attackers cannot compromise this protocol.
3. Retrieving the trusted geo-location. If the secure session establishment succeeds, the Cloud Agent logs in the TGS and the TGS starts a periodic attestation(d) Attestation of the trustworthy evidence where he TGS verifies an AIK certificate of the mobile device with the Privacy CA certificate.

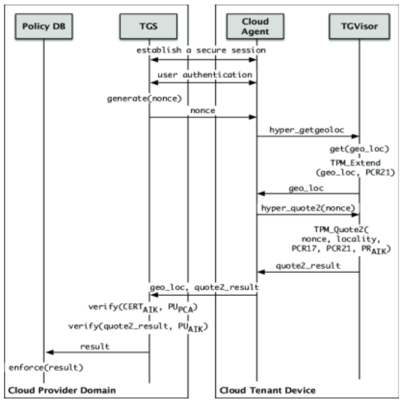


Figure 39. TGVisor Attestation Protocol

5.3.3.c TGVisor Framework Evaluation

As far as the implementation is concerned the proposed framework consists of four components: TGVisor, the TGS, the Cloud Agent, and the cloud app server. TGVisor is implemented based on eXtensible Hypervisor Framework (XMHF). XMHF delivers a general framework for building a DRTM-based tiny hypervisor. Four components are implemented:

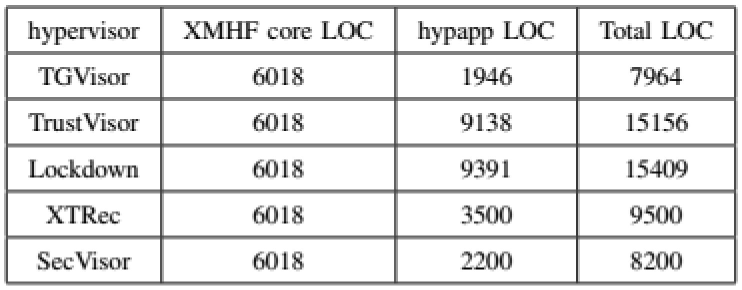
1. A core
2. A TPM
3. The TGM
4. The Hypercall Module.

The core component features memory allocation, copy data from or to an untrusted OS, and so on. The TPM component is responsible for all the TPM operations in the hypervisor such as TPM_Quote2 and TPM_Loadkey2. The TGM acquires a current geo-location of the mobile cloud device from the GPS module and provides the remote attestation value which is cryptographically protected by the TPM. The Hypercall Module features interfaces to the Cloud Agent via hypercalls.

The Cloud Agent is a lightweight program to help communicate with the TGS. The TGS is a server in the cloud provider domain that periodically attests to the TCB and the geo-locationof the mobile cloud client. The Cloud Agent and the TGSare written in Node.js. and an additional hypercall library using C language, because Node.jscannot directly invoke hypercalls. The cloud app server is a modified Etherpad server. A HP ProBook 6555b notebook is used as a mobile client device which is equipped with an AMD Turion P520 2.30GHz processor, 4GB of memory, and 160GB of HDD. The host operating system is the 32-bit version of Ubuntu 12.04.

For the evaluation of the proposed framework several tests were executed in a specific university area (geofence) where the user cloud device could not access the documents in the cloud server when it was outside of the geofence, due to illegal geo-location. On the other hand it could only access the documents when it was located inside the specific area.

The results indicate that cloud users hardly feel the performance degradation due to TGVisor. The added lines of code in TGVisor were significant less that other supervisors e.g. TrustVisor, LockDown, SecVisor etc. as a result to minimize the TCB operations. Some limitations of the proposed framework are that if the framework is to be used not only outside but indoors then the TGM need modifications in order to communicate with an indoor positioning device. Another limitation of the proposed framework is computation overhead due to TPM operations.



hypervisor	XMHF core LOC	hypapp LOC	Total LOC
TGVisor	6018	1946	7964
TrustVisor	6018	9138	15156
Lockdown	6018	9391	15409
XTRec	6018	3500	9500
SecVisor	6018	2200	8200

Figure 40. TGVisor comparison with other HyperVisors

Another experiment was the measurement of the performance impact of TGVisor with SunSpider. Since most cloud applications run in web browsers like Google Chrome, JavaScript engines are mostly embedded in them. SunSpider 1.0.2. was utilized for the TGVisor and it was compared with the native Linux system. The range of the performance impact per item is from 2.9% to 11.8%, and the average performance degradation is only 8.3% where this indicated that cloud users hardly feel the performance degradation due to TGVisor.

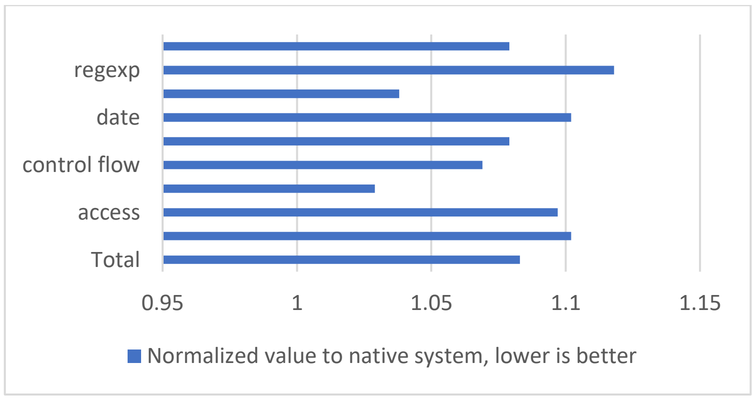


Figure 41. TGVisor Javascript performance

5.3.4 TrustVisor

A secure hypervisor, called TrustVisor, is proposed from Jonathan M. McCune, Ning Qu, Yanlin Li, Anupam Datta, Virgil D. Gligor and Adrian Perrig [44] to provide a safe execution environment for security-sensitive code modules without trusting the OS or the application that invokes the code module. TrustVisor protects security-sensitive code and data on untrusted commodity platforms from malware, e.g., kernel-level rootkits. and is designed to protect the integrity and execution of security-sensitive code, and confidentiality and integrity of the data used by that code.

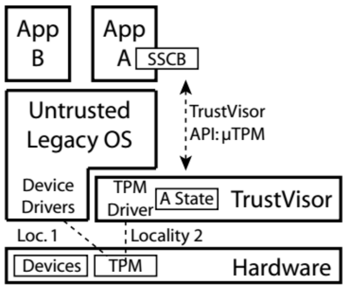


Figure 42. TrustVisor Architecture Overview

5.3.4.a Memory Protection.

TrustVisor has three basic operating modes, a host mode and two guest mode, legacy mode and secure mode. TrustVisor memory protections from the perspective of executing code. (a) In host mode, TrustVisor is executing in response to a trap or hypercall, and may manipulate the state of a PAL, or the untrusted legacy OS or applications. (b) In legacy guest mode, TrustVisor isolates PAL state and its own memory regions from the untrusted legacy code. (c) In secure guest mode, a PAL is executing, and TrustVisor isolates it from the memory regions of TrustVisor and the untrusted legacy OS and applications

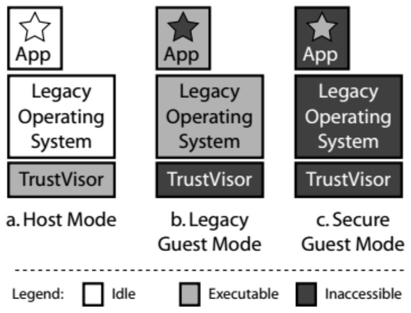


Figure 43. TrustVisor mode types

5.3.4.b Trusted Computing

A DRTM-like mechanism provides the valuable security properties of a known-good initial state, memory protection from DMA accesses, and integrity measurement of the launched code before it executes. The TRTM is realized via the inclusion of a TrustVisor-managed, software micro TPM (µTPM) instance associated with each PAL. The µTPM executes on the platform's primary CPU for high performance while avoiding the TCB growth required of a full software TPM. The TRTM is instantiated as part of the PAL registration process and is designed to serve as a "second-layer" dynamic root of trust, where the PAL code is isolated and measured before it is executed. The combination of the isolated environment, TRTM, and µTPM offer PALs facilities for fine-grained remote attestation and long-term protection of sensitive state with a small TCB.

To distinguish between legacy code and PALs, we devise a registration mechanism by which untrusted applications can register selected code and data as security sensitive. Registration triggers the sequence of TRTM operations, including allocation of a µTPM instance and protection of the PAL's memory pages. Once registered, a PAL can be invoked multiple times without requiring a new TRTM operation. The µTPM instance provides PALs with a facility for long-term secret protection and enables remote attestation that a particular PAL has executed.

TrustVisor enables remote attestation and long-term protected storage for PALs via the TRTM and µTPM associated with each PAL. TrustVisor is itself instantiated using the hardware dynamic root of trust mechanism, thereby reducing the TCB for TrustVisor and PALs executing thereupon, and rooting trust in TrustVisor in the platform's physical TPM. Figure 3 shows the relationship of trusted components when multiple PALs are registered. The shaded areas indicate the trusted components in the TCB for a particular PAL

5.3.4.c Data Secrecy

For data protection TrustVisor distinguish two intervals during which data protection is required: residence in volatile storage (RAM) while SSCB is executing, and residence on non-volatile storage while untrusted code is executing. Volatile storage refers to data in memory that is protected by TrustVisor and the system's MMU andDEV. TrustVisor-internal state is protected by keeping it in the region of memory that is accessible only to code in host mode. For Non-Volatile Storage TrustVisor utilizes cryptography by the system's Trusted Platform Module.

5.3.4.d Memory Protection Mechanisms

TrustVisor must protect its own memory regions while also isolating PALs from each other, from the legacy OS and itsapplications, and from DMA-capable devices. TrustVisor uses secure x86 hardware virtualization support to securely bootstrap itself, as well as to enforce isolation between TrustVisor itself, the legacy OS, and PALs. TrustVisor programs the system's IOMMU to prevent access to these pages by DMA-capable devices.

The life cycle of a PAL, which begins when code is first identified as comprising a PAL via a registration process is described below. PAL progresses:

PAL Registration. To avoid modifying the legacy OS to support PALs, TrustVisor implements an application-level hypercall interface for registering PALs (though PALs can also be components of the OS if desired).

PAL Invocation. Following registration, the untrusted legacy application and OS cannot read, write, or directly execute the memory containing the PAL that it registered. However, the functions inside the PAL can still be invoked using what appears to the developer to be an ordinary function call. TrustVisor then performs the following three steps before transfering control to the called function inside the PAL:
1. Identify which registered PAL contains the current called sensitive function.

2. Switch from legacy guest mode to secure guest mode, with secure guest mode configured so that only the pages containing this PAL are accessible.
3. Prepare the secure-mode execution environment for the called sensitive function. This includes marshaling input parameters into isolated pages available to the PAL and setting up the PAL's stack pointer. Passing pointers in and out of a PAL requires knowing the size of the pointed-to area. (This information is providedas part of the registration call, when entry-points are enumerated.)

PAL Termination. When a PAL has completed executing and returns to the calling legacy application, TrustVisor once again gets control. TrustVisor performs the following two steps before transfering control back to the legacy application:
1. Marshal any returned parameters and make them available to the calling untrusted application.
2. Switch from secure guest mode to legacy guest mode,in which the pages containing the PAL are once again inaccessible from guest mode.

PAL Unregistration. Un-registration is normally initiated by the application that originally registered a particular PAL. However, it can also be initiated by the legacy OS if a PAL exits due to an error (e.g., a null-pointer exception).

5.3.4.e µTPM Functions

1. The software µTPM interface exports the following TPM-like functions:
2. HV Extend for measuring data,
3. HV GetRand for getting random bytes,
4. HV Seal and HV Unseal for sealing and unsealing data based on measurements
5. HV Quote to attest recorded measurements using digital signatures.

5.3.4.f Attestation and Trust Establishment

Attestation enables a remote entity to establish trust in TrustVisor, and subsequently in PALs protected by TrustVisor. Building on the two-level integrity measurement mechanisms also design a two-part attestation mechanism. First, the TPM-based attestation to demonstrate that a dynamic root of trust was employed to launch TrustVisor with hardware-enforced isolation. Second, the µTPM-based attestation to demonstrate that TRTM was employed to launch a particular PAL with TrustVisor-enforced isolation. Thus, the ultimate root of trust in a system running TrustVisor stems from TPM-based attestation to the invocation of TrustVisor using hardware DRTM.

TPM-Generated Attestation. An external verifier that receives a TPM-generated attestation covering the PCRs into which TrustVisor-relevant binaries and data have been extended conveys the following information to the verifier: A dynamic root of trust was used to bootstrap the execution of TrustVisor. TrustVisor received control immediately following the establishment of the dynamic root of trust.

The precise version of TrustVisor that is executing is identifiable by its measurement in one of the PCRs. TrustVisor generated an identity key for its µTPM based on the current TPM AIK. Note that the verifier must learn the identity of the AIK by some authentic mechanism, such as pre-configuration by an administrator or system owner. In some cases, trust-on-firstuse may even be reasonable, but we emphasize that the choice of mechanism is orthogonal to the architecture of TrustVisor. µTPM-Generated Attestation.

An attestation from TrustVisor consists of an HV Quote operation, along with additional measurement metadata3 to facilitate the verifier's making sense out of the values in the µPCRs. The verifier must first decide to trust TrustVisor based on a TPM attestation. If TrustVisor is untrusted, then no trusted environment can be constructed using TrustVisor. A verifier learns the following information as it analyzes the contents of the µPCRs: µPCR always begins with 20 bytes of zeros extended with the measurement of the registered PAL.

Thus, the verifier can learn precisely which PAL was registered and invoked during this session on TrustVisor. The values in the remaining µPCRs and any other values extended into µPCR [0] are specific to the PAL that executed and will not have been influenced by TrustVisor. The set of µPCRs selected for inclusion in HV Quote will be signed by TrustVisor's µTPM identity key µAIK.

5.3.4.g TrustVisor Framework Evaluation

For the implementation memory protection mechanisms are executed, then trusted computing mechanisms including the µTPM implementation. To achieve memory isolation, TrustVisor virtualizes the guest OS's physical memory using the 2D nested page table (NPT) hardware feature provided by AMD SVM. The NPTs are maintained by TrustVisor in host mode, while the guest OS continues to maintain its own page tables to translate guest virtual addresses to guest physical addresses (i.e., the guest OS need not be aware that it is virtualized).

At runtime, guest physical addresses are further translated to machine physical addresses by the CPU using the corresponding NPT. TrustVisor maintains only one set of NPTs for the guest, which is simply an identity mapping from guest physical addresses to machine physical addresses. TrustVisor uses 2 MB page granularity in the NPTs to improve performance by reducing TLB pressure. To protect itself, TrustVisor sets the NPT permissions such that its physical pages can never be accessed through the NPT from guest mode.

To protect its physical pages against DMA access by devices, TrustVisor uses the DEV (Device Exclusion Vector) mechanism, which is a simplified IOMMU (Input Output Memory Management Unit) provided by AMD SVM. With DEV support, the system's memory controller is designed to provide DMA read and write protection for physical pages on a per-page basis. Application developers must explicitly register and unregister the PAL(s) for their application (recall §4.2.2).

Both registration and unregistration consist of a hypercall with parameters to describe the PAL to be registered. These hypercalls are intercepted directly by TrustVisor without legacy OS awareness using the VMMCALL instruction. Finally, the trust computing mechanisms are executed: Trust booting where AMD's SKINIT instruction is used to create a dynamic root of trust to bootstrap TrustVisor starting from an initially untrusted system state and then the μTPM implementation.

The experimental platform was a Dell PowerEdge T105 with a Quad-Core AMD Opteron running at 2.3 GHz. The current implementation of TrustVisor allocated 2 GB of RAM to the Linux kernel and supports only a uniprocessor guest. Additional cores and RAM were unused. The server run the 32-bit version of the Fedora Core 6 Linux distribution for the experiments.

It was evaluated how this implementation maintains a small TCB and compatibility with unmodified legacy software. Results showed that the TCB was reduced, the total size of TrustVisor implementation is 7889 lines of C and assembly code and the runtime TCB was about 6481 lines, which includes 3919 lines of RSA and other libraries. TrustVisor can support any 32-bit legacy x86 OS image without any modifications. TrustVisor has an extra overhead due to PALS and μTPM operations.

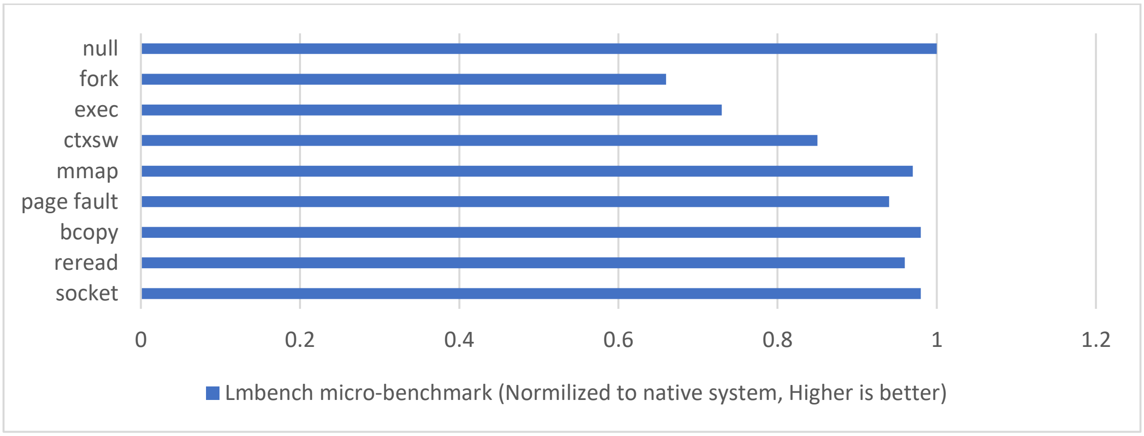


Figure 44. Comparison between TrustVisor and Linux Native.

5.3.5 SecVisor

SecVisor is a proposed framework from Arvind Seshadri, Mark Luk, Ning Qu and Adrian Perrig[ ]where is a tiny hypervisor that uses hardware memory protection and memory virtualization and ensures code integrity for commodity OS kernels [45]. In particular, SecVisor ensures that only user-approved code can execute in kernel mode over the entire system lifetime, as a result to protect the kernel against code injection attacks, such as kernel rootkits.

5.3.5.a SecVisor Framework Overview

SecVisor uses the IO Memory Management Unit (IOMMU) to protect approved code from where Direct Memory Access (DMA) writes. Also, SecVisor virtualizes the CPU's Memory Management Unit (MMU) and the IOMMU. This ensures that SecVisor can intercept and check all modifications to MMU and IOMMU state. The SecVisor ensures that CPU executes only approved code in kernel mode. Every entry into kernel should set the CPU's Instruction Pointer (IP) to an instruction within approved kernel code. After an entry into kernel mode places the IP within approved code, the IP should continue to point to approved kernel code until the CPU exits kernel mode. Every exit from kernel should set the privilege level of the CPU to user mode. SecVisor ensures that the approved code can be only modified by SecVisor and its TCB.

SecVisor uses page tables as the basis of its hardware memory protections. SecVisor can keep the page tables in its own address space and allow the kernel to read and modify them only via "safe" function calls. Also, SecVisor virtualizes physical memory. Virtualizing physical memory causes the addresses sent on the memory bus to be different from the physical addresses seen by the kernel. The page table used by SecVisor to virtualize physical memory is called Protection Page Table. SecVisor sets the Protection Page Table so that user memory is not executable when the CPU executes in kernel mode.

On each entry to kernel mode, SecVisor sets execute permissions in the Protection Page Table so that only approved code will be executable. Then, the CPU will generate an exception on every attempt to execute unapproved code in kernel mode. When SecVisor receives such an exception, it terminates the kernel. SecVisor also marks the approved code pages read-only in the Protection Page Table. This prevents any code executing on the CPU (except SecVisor) from modifying approved code pages. SecVisor uses the DMA write protection functionality of the IOMMU to protect approved code pages from being modified by DMA writes.

SecVisor ensures that all control transfers through which the CPU enters kernel mode will set the IP to an address within the approved code. This requires SecVisor to find the target of every possible control transfer to kernel mode is that CPUs only allow kernel mode entries to transfer control to entry points designated by the kernel. This prevents user programs from triggering arbitrary control flows in kernel code by entering at arbitrary points. The kernel informs the CPU of the permitted entry points by writing the addresses of such entry points (hereafter called the entry pointers) in CPU registers and data structures like the interrupt vector table (IVT).

Then, SecVisor only has to ensure that all entry pointers point to instructions within approved code. To find all the entry pointers, it needs to identify all the CPU data structures that can contain entry pointers. By design, every CPU architecture has a set of control transfer events that trigger CPU execution privilege changes. Each control transfer event has an associated entry pointer in some CPU data structure.

The entry list can be created from the architectural specification of the CPU. Next, for each event in the entry list we find the CPU data structure which holds its entry pointer. In this manner, we obtain the list of all the CPU data structures which can hold the entry pointers. SecVisor virtualizes the entry pointers and only permits the kernel to operate on the virtualized copies. This allows SecVisor to intercept and check all modifications to the entry pointers.

The virtualization can be performed in two ways. First, SecVisor can provide the kernel with “safe” function calls through which the kernel can read and modify the entry pointers. Second, SecVisor can maintain shadow copies of the entry pointers for use by the CPU and keep the shadow copies synchronized with the kernel’s entry pointers. As with virtualizing physical memory, the choice between these two alternatives is a trade-off of performance versus security and portability.

The shadowing method was preferred in this framework because it reduces the size of SecVisor’s kernel interface and also reduces the number of changes required to port a kernel to SecVisor. All legitimate methods that exit kernel mode will transfer control to code in user memory. If on each entry to kernel mode the CPU will start executing approved code, it is fairly direct to ensure that exits from kernel mode will set the CPU privilege to user mode. All kernel mode entries will try to execute approved code, which is part of kernel memory. This will cause the CPU to generate an exception. As part of handling this exception, SecVisor marks all user memory non-executable. Thus, any exit to user mode will cause a protection violation, generating a CPU exception. SecVisor sets the privilege level of the CPU to user mode.

5.3.5.b SecVisor Framework Evaluation

For the evaluation of SecVisor three design goals where considered: small code size,minimal kernel interface, and ease of porting OS kernels. For the Code size it was utilized a D.A. Wheeler’s sloc program to count the number of lines of source code SecVisor prototype, the Kernel interface was consisted of only 2hypercalls. The first hypercall was used by the kernel to request changes to its code (such as loading and unloading modules), while the second hypercall was used by the kernel during its initialization to pass the virtual and guest physical addresses of the shadow table area and for porting OS kernels three changes to the Linux kernel version 2.6.20 to port it to SecVisor. First, the decompress_kernel function invokes SecVisor using the skinit instruction instead of jumping to the decompressed kernel. Second, during its initialization, the kernel passes the addresses of the shadow table area to SecVisorusing a hypercall.

Finally, the control flow of the load_module and the free_module function was changed.The experimental platform was a HP Compaqdc5750 Microtower PC. This PC uses an AMD Athlon64 X2 dual core CPU running at 2200 MHz and has 2 GB RAM. SecVisor allocates 1536 MB of RAM to the kernel in the experiments. The PC runs the i386 version of the Fedora Core 6 Linux distribution. We use the uniprocessor versions of Linux kernel 2.6.20 and Xen 3.0.4. The lmbench benchmarking suite was used to measure overheads of different kernel operation.

The results showed SecVisor protects the kernel against a variety of well-known and unpublished attacks, including code injection through buffer overruns, kernel-level rootkits, and malicious devices with DMA access but does not prevent against control-flow attacks, it can be combined with approaches that do provide additional protections. SecVisor will ensure code integrity and memory protection.

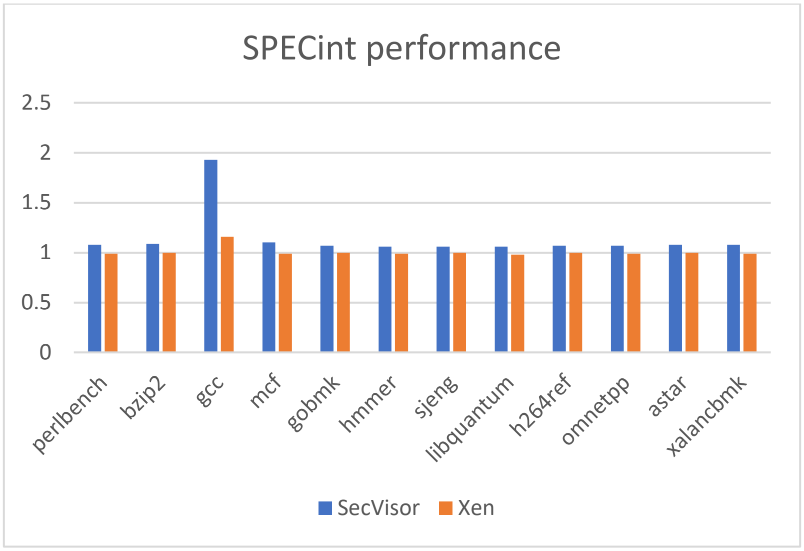


Figure 45. SecVisor VS Xen performance

5.3.6 Lockdown

Red/green systems have been proposed as a mechanism for improving user security without abandoning the generality that has made computers so successful. They are based on the observation that users perform security-sensitive transactions infrequently, and hence enhanced security protections need only be provided on demand for a limited set of activities [46]. They require virtualizing all of the system resources and devices that may be shared between the two environments. From a security perspective, this introduces considerable complexity into the reference monitor responsible for keeping the two environments separate. In addition, even without compromising a reference monitor, actively sharing resources by allowing both environments to run simultaneously exposes side channels that can be used to learn confidential information. From a performance perspective, the interposition necessary to virtualize devices adds overhead to both trusted and untrusted applications.

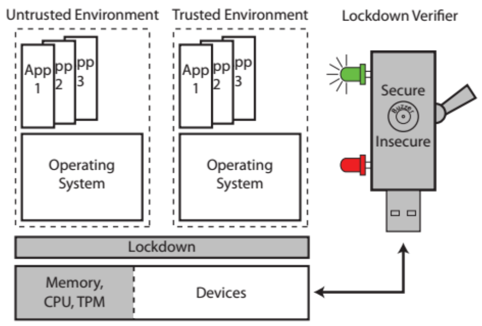


Figure 46. Lockdown Overview

At a high level, Lockdown splits system execution into two environments, trusted and untrusted, that execute non-concurrently. This design is based on the belief that the user has a set of tasks that he wants to run with maximum performance, and that he has a set of tasks that are security sensitive which he wants to run with maximum security and which are infrequent and less performance critical. The performance-sensitive applications run in the untrusted environment with near-native speed, while security-sensitive applications run in the trusted environment, which is kept pristine and protected by Lockdown.

The Lockdown architecture is based on two core concepts: (i) hyper-partitioning: system resources are partitioned as opposed to being virtualized. Among other benefits, this results in greater performance, since it minimizes resource inter-positioning, and it eliminates most side-channel attacks possible with virtualization; and (ii) trusted environment protection: Lockdown limits code execution in the trusted environment to a small set of trusted applications and ensures that network communication is only permitted with trusted sites.

5.3.6.a Hyper Partitioning

As far as hyper-partitioning is concerned, Lockdown must isolate the trusted environment from the untrusted environment. Further, Lockdown must isolate itself from both environments so that its functionality cannot be deliberately or inadvertently modified. One way to achieve this isolation is to rely on the platform hardware to partition resources. This hardware capability facilitates concurrent execution of multiple partitions without virtualizing devices but not all devices can be share and such platform support is not widely available. Lockdown partitions the CPU in time by only allowing one environment to execute at a time. With hyper-partitioning, both the untrusted and trusted environments use the same set of physical devices and leverages the Advanced Configuration and Power-management Interface (ACPI) to save and restore device states while partitioning non-storage devices. Lockdown performs an environment switch by transitioning the current environment to sleep and waking up the other. Lockdown uses approved code execution and network protection to ensure that only trusted code (including device firmware code) can be executed and only trusted sites can be visited while in the trusted environment, as explained below.

5.3.6.b Approved Code Execution

For non-firmware code, Lockdown uses Nested Page Tables (NPT) to enforce a W $\oplus$X policy on physical memory pages used within the trusted environment. Thus, a page within the trusted environment may be executed or written, but not both. Prior to converting a page to executable status, Lockdown checks the memory region against a list of trusted. Execution is permitted only if this check succeeds.

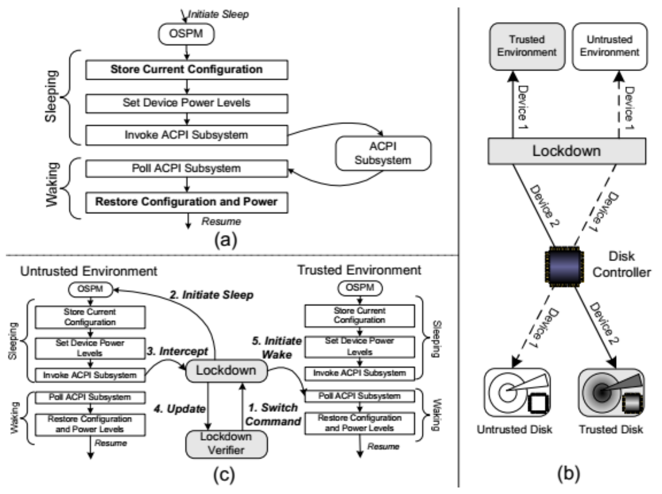


Figure 47. Lockdown protection mechanism

5.3.6.c Network Protection

Since users perform many security-sensitive activities online, applications executing in the trusted environment need to communicate with remote sites via the network. However, permitting network communication exposes the trusted environment to external attacks. Remote attackers may exploit flaws in the OS’s network stack, or the user may inadvertently access a malicious site, or a network-based attacker may perform SSL-based attacks (e.g., tricking a user into accepting a bogus certificate).While approved code execution prevents many code-based attacks, the trusted environment may still be vulnerable to script-based attacks (e.g., Javascript) and return-oriented programming attacks.

To forestall such attacks, Lockdown restricts the trusted environment to communicate only with a limited set of trusted sites. It imposes these restrictions by inter posing on all network traffic to or from the trusted environment. Lockdown uses hardware CPU and physical memory protections to prevent the trusted environment from seeing or accessing any physical network devices present in the system.

Network communication is permitted via a proxy network driver that Lockdown installs in the guest OS. This driver forwards packets to Lockdown, which analyzes the packets and then forwards them to the physical network interface. The trusted environment can use a distinct physical network interface or reuse the same interface of the untrusted environment for network communication (since the environments run non-concurrently). In both cases the Lockdown hypervisor will need to include the network driver for the physical interface.

A simpler approach is to perform network access (either wireless or wired) using the Lockdown Verifier. In this case, the Lockdown hypervisor does not need to contain any network driver but simply forwards the packets to the verifier. Lockdown uses packet analysis to determine which network packets are permitted. One approach, with the argument that any site with sensitive data should be using SSL to protect it in transit, would be to allow only SSL and DNS network packets to passthrough to trusted sites. All other packets are dropped.

When an SSL session is initiated, Lockdown determines if the request is a valid SSL connection request. If it is, Lockdown validates the site’s SSL certificate and checks it against the list of trusted sites (the creation and maintenance of this list is discussed in the following section). If any of these checks fail, the packet is dropped. Incoming packets are permitted only if they belong to an existing SSL session or are in response to an earlier DNS request. Note that DNS-based attacks are forestalled by SSL certificate verification. From a technical perspective, supporting other network protocols such as SSH is also possible.

5.3.6.d Defining Trusted Entities.

To keep the trusted environment safe, Lockdown restricts the software that can execute and the sites that can be visited. To define what software and sites can be trusted, we leverage the user’s existing trust in the distributor of Lockdown, i.e., the organization that provided the user with a copy of Lockdown in the first place. For example, in a corporation, the IT department would play the role of Lockdown distributor.

For consumers, the role might be played by a trusted company or organization, such as RedHat, Mozilla, or Microsoft. Lockdown’s key insight is that by agreeing to install Lockdown, the user is expressing their trust in the Lockdown provider, since Lockdown will be operating with maximum platform privileges on their computer. Thus, we can also trust that same organization to vet trusted software and websites. The list of trusted software can be relatively small: primarily an operating system and a trusted browser.

5.3.6.e Lockdown Framework Evaluation

The implementation is a complete prototype of Lockdown on both AMD and Intel x86 platforms with Windows 2003 Server as the OS in both the trusted and untrusted environments. It was also developed a prototype using Linux guests. Neither prototype required changing any code in the OS kernels. Due to space constraints, we focus on describing our Windows prototype on the AMD platform. This Lockdown prototype consists of a Lockdown Loader and the Lockdown Runtime. The SKINIT instruction is used to perform a late-launch operation which ensures that the Lockdown Loader runs in a hardware-protected environment and that its measurement (cryptographic hash) is stored in the TPM’s Platform Configuration Register (PCR) 17.

The trusted Lockdown Loader loads the Lockdown Runtime and protects the Lockdown Runtime’s memory region from DMA reads and writes (using AMD’s Device Exclusion Vector). It then verifies the integrity of the Lockdown Runtime and extends a measurement (a cryptographic hash) of the Lockdown Runtime’s code into the TPM’s PCR 19. The Lockdown Loader then initializes the USB controller on the host for communication with the Lockdown Verifier, creates the Nested Page Tables for the trusted and untrusted environments and transfers control to the Lockdown Runtime.

When first launched, the Lockdown Runtime requests a challenge from the Lockdown Verifier. The Lockdown Runtime launches the environment currently indicated on the Lockdown Verifier in a hardware virtual machine and informs the Lockdown Verifier once the environment has been launched, so that the Lockdown Verifier can sound the attention buzz and light the appropriate LED. The Lockdown Runtime's role in hyper-partitioning, and protection of the trusted environment is described below.

To implement hyper-partitioning for non-storage devices under the Windows OS, Lockdown makes use of the ACPI S4 (hibernate) sleep state. ACPI S3 (standby) would offer faster switching times, but Windows ACPI implementation only saves and restores device state during an S4 sleep, and hence we cannot use S3 with Windows without modifying its source code. Memory and storage device partitioning are described below. Memory. In our current implementation (on systems with 4 GB of physical memory), Lockdown reserves 186 MB for itself and 258 MB for the system's firmware. The rest of physical memory is available to the trusted or untrusted environment. Storage Devices.

To implement the Trusted Environment, Lockdown uses page level code hashing. Prior to executing the trusted environment, Lockdown sets its Nested Page Table (NPT) entries to prevent execution of those pages. When the trusted environment attempts to execute a page, it causes a fault that returns control to Lockdown. Lockdown computes a hash of the faulting page and compares it to the hashes in its list of trusted software. If a match is found, the corresponding NPT entry is updated to allow execution but prevent writes. If the trusted environment later writes to this page, a write fault will be generated. Lockdown will re-enable writing but disable execution. Network Protection. To provide network protection for the trusted environment, an untrusted network was developed driver for Windows, and an SSL Protocol Analyzer within Lockdown.

Lockdown's TCB compares favorably with other popular hypervisors and VMMs ,such as L4Ka-Pistachio, NOVA, VMWare ESXi, Xen + Linux, KVM + Linux + QEMU and Hyper-V + Windows which tend to be orders of magnitude larger, despite not providing Lockdown's protection's for a trusted environment The implementation and results indicate that partitioning offers increased security (by reducing the size of the reference monitor to 10K lines of code and by reducing opportunities for side channels) and performance (by giving the untrusted environment unfettered access to system devices) at the cost of slow switching times (on current systems).

5.3.7 Credo

Credo is a Hyper-V based hypervisor [47]. Key components of the Hyper-V architecture are a microkernel hypervisor and a privileged management partition, called the *root partition*. The hypervisor virtualizes core platform resources while the root partition owns all I/O devices and does I/O virtualization. Hyper-V supports two forms of virtualized I/O – *emulation based I/O*, where a guest VM performs memory mapped or port-based I/O that is intercepted by the hypervisor and forwarded to the root partition and *enlightened*

I/O, where the root partition and a guest VM communicate over a shared-memory channel using the *vmbus* protocol.

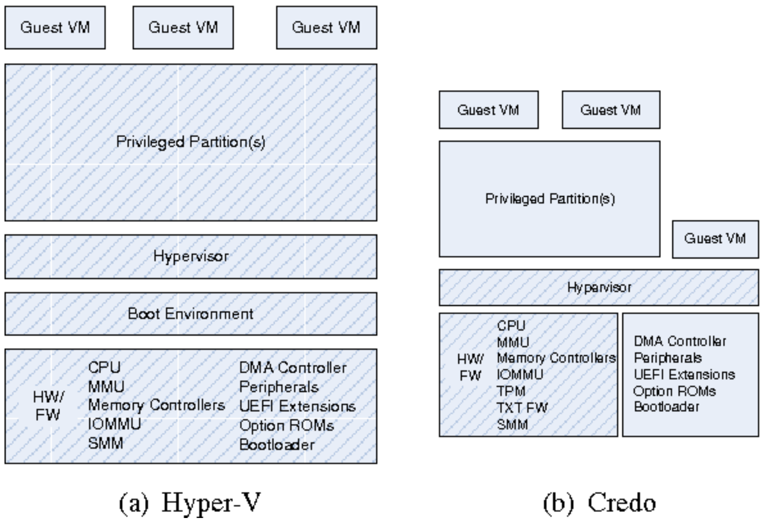


Figure 48. Credo comparison with Hyper-V

The Credo threat model does not consider certain types of attacks. In particular the primary goals of Credo is to provide secrecy and integrity protection of a VM's virtualized state from the root partition, establish a small, measurable TCB for a VM as well as a measure of VM's trustworthiness, and enable mechanisms to verify this at runtime and to make sure that the cost of security should be imposed only if security is required by a VM. Since performance is the key requirement in any cloud computing environment, any such cost should be low.

5.3.7.a Credo Framework Overview

The Credo architecture provides a way to execute guest virtual machines in a secure and trustworthy environment without taking a trust dependency on the root partition using a mechanism similar to DRTM launch. The hypervisor provides a hypercall to trigger a *v(irtual) DRTM* event for a guest VM. When this event is triggered (either by the guest VM or by the root partition on its behalf), the hypervisor suspends the VM, creates the secure execution environment for the VM using the *emancipation* procedure, *measures* and *records* the "execution state" of the VM. As the last step, the hypervisor resumes the execution of the guest VM inside the secure execution environment.

To emancipate a guest VM's memory, the hypervisor removes root partition's access from its page tables for all system memory pages backing a guest VM's physical address space. Both reads and writes to these pages are intercepted by the hypervisor - reads return all 0*xFF* s, while writes are silently thrown away. After the VM is emancipated, the hypervisor disallows creation of any new mappings in an emancipated VMs physical space. This implies that that a guest VM's address space must be completely populated before the VM is emancipated.

However, this does not preclude dynamic memory management where the VM can unemancipate memory pages before returning them to the root partition. Because an emancipated VM has exclusive control over memory, it must explicitly release control of the memory pages backing its physical space in order for the resources to be reclaimed by the root partition after the VM shuts down. This step is accomplished using the “unemancipate partition” hypercall, which resets the root partition’s access to its original state for all memory pages backing the guest VM.

It is imperative that the guest VM must explicitly remove any secret information from pages explicitly before calling the unemancipate partition hypercall in order to maintain the secrecy and integrity guarantees. A guest VM’s vCPU state may be modified outside the control of the guest VM as a result of *intercepts*. These intercepts are either caused by guest VM itself, e.g., by accessing some virtual resource such as MSR or I/O port, or by external events, such as a virtual interrupt associated with a virtual device

5.3.7.b Emancipating I/O

In Credo, it is the responsibility of the guest VM to use *cryptographic* measures for I/O emancipation as the hypervisor is not involved in the para virtualized I/O path. Making the guest VM aware of I/O emancipation is compatible with the para virtualized I/O model. Emancipated para-virtualized I/O from a guest VM involves two steps: first, a shared memory based channel is established between an emancipated guest VM and the untrusted root partition; and second, a guest VM uses secrecy and/or integrity protection techniques to read or write data to or from this shared memory channel. For the root partition to use shared memory for communication it needs to be able to access guest memory which is by default protected by memory emancipation.

Credo provides the “unemancipate page” hypercall to selectively remove protection for the pages used by the shared memory channel. One such channel is created for each para-virtualized device. Messages sent over the channel may contain pointers to buffers on data pages that must also be unemancipated. As an optimization, instead of calling the “unemancipate page” hypercall for every page, the vmbus keeps a pool of unemancipated pages that are setup at VM startup.

Drivers allocate and free memory pages to and from this pool. The vmbus driver in the guest VM can grow/shrink this pool on demand as needed. This encryption approach to emancipating I/O works in a cloud environment since the guest VM mostly requires just storage and network based I/O to execute in such an environment. In fact, Credo explicitly disallows any emulation based I/O, and all VM management should be performed using a network based remote access connection.

5.3.7.c Credo Framework Evaluation

(a) Emancipated VM startup:

In order to build such an “execution image” that is formed from a saved VM state that forms the captured via VM save operation a bootable RAMDISK is stored on a virtual IDE disk. Here, the IDE disk is only used by the bootloader on the trusted server. By disabling the IDE disk driver (disk.sys) from this RAMDISK installation, is ensure that once Windows finishes booting, the OS does not use the IDE disk with which the VM started executing.

(b) SDisk:

SDisk is implemented as a filter driver in the virtual storage stack that sits right above the virtual SCSI driver. It mainly interposes itself on the read/write path, and performs data encryption/decryption before passing it down to virtual Encryption mechanism used is AES with 256-bit key as KSD. SCSI driver, which then sends emancipated I/O data on untrusted vmbus communication channel. SDisk driver stores metadata related to SDisk on the last 1MB of vhd.

(c) DRTM launch of hypervisor:

TXT architecture puts certain restrictions on the DLME that make it difficult to directly launch the hypervisor as a DLME. Instead a small Hyper-V aware DLME (HvDLME) that understands the specifics of Hyper-V hypervisor is used as DLME. It extends Windows bootloader (winload.exe) to load HvDLME and the actual hypervisor binary in the memory and perform a small amount of DRTM specific configuration to establish the required memory mappings, e.g., to allow TPM access. Next, the boot loader launches HvDLME using DRTM launch as described earlier. As a result of DRTM operation, PCRs 17 and 18 are set with measurements related to HvDLME.

The PassMark Performance Test benchmark suite [] was used for benchmarking the performance of various tests inside the guest VM. The benchmark is run in three scenarios: With stock Hyper-V configuration. With Credo but without the secure execution environment. This measures the impact of Credo on no security sensitive VMs. With Credo within the secure execution environment.

This measures the impact of Credo and secure execution environment on security sensitive VMs. Experimental results show that Credo imposes mostly one-time setup cost. Credo does not impact performance for virtual machines that do not require the security benefits when compared to a stock Hyper-V environment, while only imposing modest cost on emancipated VMs.

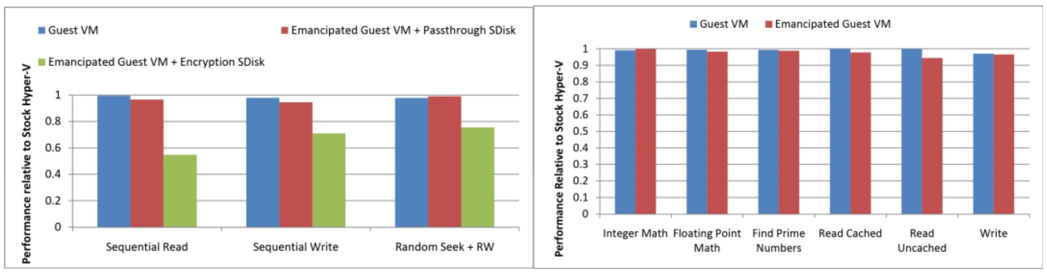


Figure 49. Credo Performance Evaluation CPU/Memory and Disk

5.3.8 Hypervisors Summary

In this chapter we presented some representative virtualization frameworks alongside with their implantation and instruction for execution and configurations. We present the hypervisor information summary in the table below.

Each row contains the name of each hypervisor followed with the contributors, date of publication, if the hypervisor uses a TPM, the supported operation systems, if the framework has been tested by the contributors and if there is the source code available for downloading.

Table 8. Summary of the hypervisors that are presented in this paper.

Framework Name	Contributors	Date of publication	TPM Version	Type	OS	Evaluation	Source Code
XMHF	**[40]**	June 26, 2012	TPM (v1.2)	1	a)Windows XP b)Windows Server 2003 c)Ubuntu 10.04	The framework was tested	In Github. Open Source in official site
uberXMHF	**[41]**	October 3 2018	TPM (v1.2 or above)	1	a)Ubuntu 16.04 b)Ubuntu 12.04 c)Raspberry PI 3	The framework was tested	In Github. Open Source in official site.
XVisor	**[42]**	January 31 2016	No TPM	1	Linux Raspberry Pi	The framework was tested	In Github. Open Source in official site.
TGVisor	**[43]**	June 2015	TPM usage	1	Ubuntu 12.04	The framework was tested	Not Found
TrustVisor	**[44]**	May 2010	TPM (v1.2 or above)	1	Fedora Core 6 Linux	The framework was tested	In Github.
SecVisor	**[45]**	October 14 2007	TPM usage	2	Fedora Core 6 Linux	The framework was tested	Not Found
Lockdown	**[46]**	July 14, 2009	TPM usage	1	a) Windows 2003 Server b) Linux	The framework was tested	In Github.
Credo	**[47]**	2011	TPM (v1.2)	1	Windows	The framework was tested	Not Found

6. PROPOSED SECURE GEO-LOCALIZATION FRAMEWORK

In this chapter we present our proposed Localization framework that ensures a trust geo-location environment in WSN. The framework is explained with details alongside with detailed explanation for the way that the proposed framework achieves trust geo-location. Geographic locations of user devices are widely used to provide rich user experience in various environments such as proximity-based marketing, travel information, and cloud computing. Especially, cloud service providers require to utilize actual cloud user's locations in location-based cloud services like Amazon GovCloud.

However, it is not trivial to obtain the trusted geo-locations of the user devices because there are many points for attackers to forge the current geo-locations of the cloud user devices. WSN may be deployed in hostile environments with sensors operating unsupervised. Attacks on WSN are presented in section 4. Hence, an adversary can interrupt the functionality of location-aware applications by exploiting the vulnerabilities of the localization scheme. To confront those types of attacks we proposed a Secure Geo-Location framework with the utilization of Hypervisors and TPM.

6.1 Introduction

In this section our proposed secure geolocation framework is presented, with the usage of a Trust Platform Module (TPM) and a Hypervisor, that ensures a safe geolocation process for unknown nodes in a specified and trust environment. The basic idea is to create a geographical environment that is consisted of anchor nodes and where unknown node can be located and can access a protected and safe system.The basic concept of the framework is as follows:

First, we specify a geographical field that will be the environment where geolocation and safety will be provided. This field consists of a base station and three anchor-beacon nodes that their geological position is calculated and known. Those nodes form a tetrahedron shape (the base station and the two anchor nodes form the base and one node is the pick). Each node supports a Hypervisor that can ensure a trustful and safe environment. In our implementation we use the Trustrvisor (which is implemented over a uberXMHF), further details are presented below.

The TPM is utilized here to achieve this purpose since it provides all the mechanisms of a secure crypto processor designed to secure hardware through integrated cryptographic keys. When an unknown node enters the specified field, it can easily be detected from all four nodes. With a localization algorithm, the coordinates of the unknown node can be calculated (further details of this process will be presented in the next session) and if the unknown node position is determined as then the access to the system is granted, otherwise is denied. In the case that is granted the anchor node can give access to the new node in the system through a Hypervisor (Virtual Machine).

With the hypervisor the node can utilize all the information in the system without data corruption. Thus, the new node has access to a secure system that protects it from any malicious attacks, also the system is protected from any possible attempt in data modification, data corruption or any other type of attacks in WNS. Below a diagram is presented that explain the high-level overview of the basic concept of our framework. The framework contains four phases:

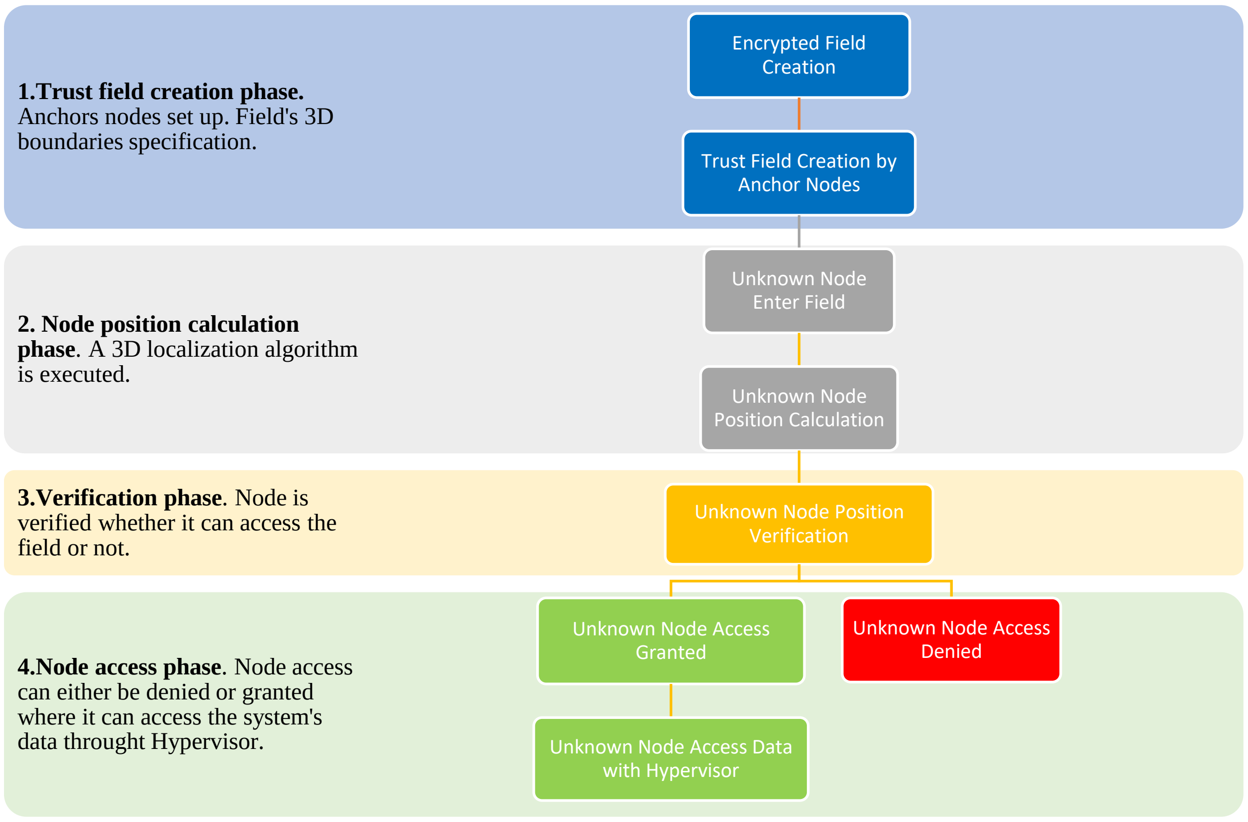

1.Trust field creation phase. Anchors nodes set up. Field's 3D boundaries specification.
Encrypted Field Creation
Trust Field Creation by Anchor Nodes
2. Node position calculation phase. A 3D localization algorithm is executed.
Unknown Node Enter Field
Unknown Node Position Calculation
3.Verification phase. Node is verified whether it can access the field or not.
Unknown Node Position Verification
4.Node access phase. Node access can either be denied or granted where it can access the system's data throught Hypervisor.
Unknown Node Access Granted
Unknown Node Access Denied
Unknown Node Access Data with Hypervisor

Figure 50. Proposed framework high level overview.

Trust field creation

A three-dimensional geographical field is formed by three coplanar and static base nodes on the ground, forming a triangular shape. A fourth node stays on a specific high position from the ground as a results all four nodes are forming a tetrahedron shape field. All four nodes have known coordinates that are shared between them and are in a constant communication.

Node position calculation

When an unknown node enters this specified trust field, it can be detected from all the base nodes of the system. Then the node’s distance can be estimated using positioning techniques (e.g. RSSI, DV-HOP etc.) and its position inside the system is calculated using relevant techniques (e.g. trilateration etc.). Those coordinates are shared between the base nodes.

Verification phase

Knowing the node's specific position compared with the system the base nodes can estimate if the node is inside or outside of the trust field. Then the system can accept or reject the node access in the system's data.

Node access phase

If the node is not rejected from the system, its access is granted. Then the hypervisor (e.g. Hypervisor) is responsible of creating a safe environment (a guest virtual machine) for the node, where the node has access to the system's data and can interact with it.

6.2 Framework Overview

The framework consists of the following features:

- Four (or five) base nodes, forming a tetrahedron shape A, B, C and D. (E).
- Each node contains a hypervisor.
- Each node contains a TPM for the hypervisor implementation.
- A unknow node X enters the field with a random movement.

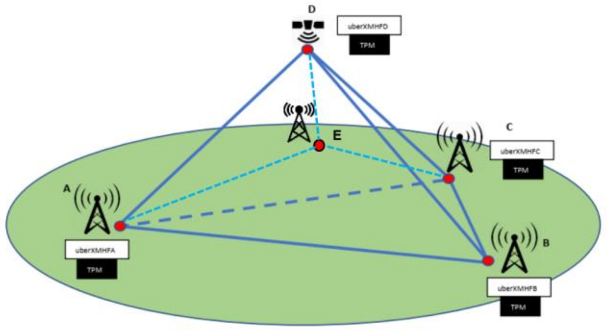


Figure 51. Framework overview

6.2.1 Trust Field Creation

In figure 53 we can understand how this specified three-dimensional field can be represented in a cartesian plane. Nodes A, B, (E) and C are base ground nodes on a coplanar field. Node D has a known high H from the ground. In the cartesian plane representation the base nodes form the tetrahedron ABCD with base A, B, (E), C and pick D. The edges of the tetrahedron represents the distances between the node, that are also known and constant: d1 is distance between A and B, d2 is distance between B and C, d3 is distance between A and C, d4 is distance between A and D, d5 is distance between C and D and d6 is distance between B and D etc. Each base node utilizes a hypervisor in order to create a safe and secure environment for the new guest nodes: uberXMHFA for node A, uberXMHFB for node B, uberXMHFC for node C, uberXMHFE for node E and uberXMHFD for nodeD. In our framework we choose the Trustvisor hypervisor which is implemented from uberXMHF. Each node contains also a TPM environment, which is necessary for the Trustvisor hypervisor in order to create a safe environment. All the system's geographical data, base node's coordinates and distances, are known and shared between all base nodes. All base nodes are in a constant communication and aware of any change in the system.

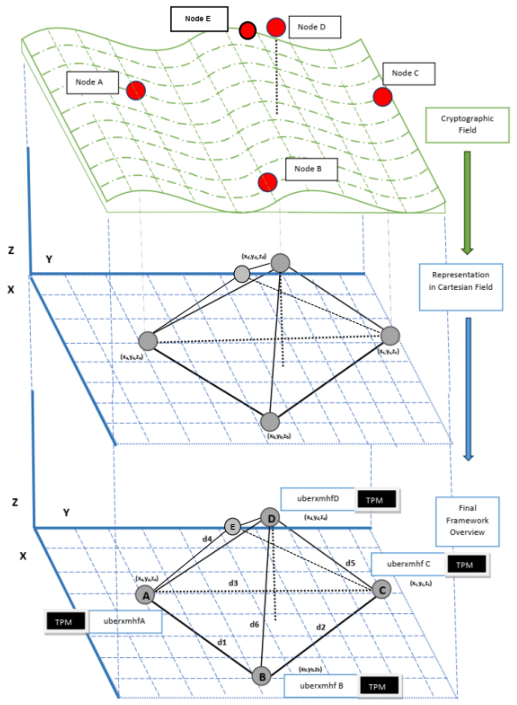


Figure 52. Trust field representation.

6.2.2 Node Position Calculation

In order to calculate the coordinates of the unknown node X in the system we utilized the mechanisms below:

Table 9. Localization algorithms simulated

Algorithm Name	Metrics	Dimension	Base nodes
1.Multilateration	RSSI	3	4
2.Multilateration	TOA	3	4
3.Direct Location Method	RSSI	3	4
4.Direct Location Method	TOA	3	4
5.CHAN Algorithm	TDOA	3	5
6.Hybrid Algorithm	TDOA/ TOA	3	4

6.2.2.a 3D-RSSI/TOA

Finding Distances from center points from 3 spheres

In order to calculate the distances D1, D2 and D3 of the unknown node X, we need a minimum of 3 base nodes in the area. The selected techniques for this calculation are: RSSI and TOA. All three localization techniques can calculate the distances between two nodes in a system.

In our framework, the distances D1, D2 and D3 can be calculated by those techniques. We can use every combination of three base node for the distance calculation. As a representative case we can have as a reference base nodes A, B and C (or D) and their distances from unknown node E D1, D2 and D3. The distances represent the radius of each sphere with centers nodes A, B and C

Finding intersection points between 3 spheres

We assume that the coordinates of the spheres (x_A, y_A, z_A), (x_B, y_B, z_B) and (x_C, y_C, z_C) , alongside with the relevant radius D1, D2 and D3 are known. Then we have the following equations. Three spheres:

$$\text{EQ1: } (x_A - x)^2 + (y_A - y)^2 + (z_A - z)^2 = D1^2$$

$$\text{EQ2: } (x_B - x)^2 + (y_B - y)^2 + (z_B - z)^2 = D2^2$$

$$\text{EQ3: } (x_C - x)^2 + (y_C - y)^2 + (z_C - z)^2 = D3^2$$

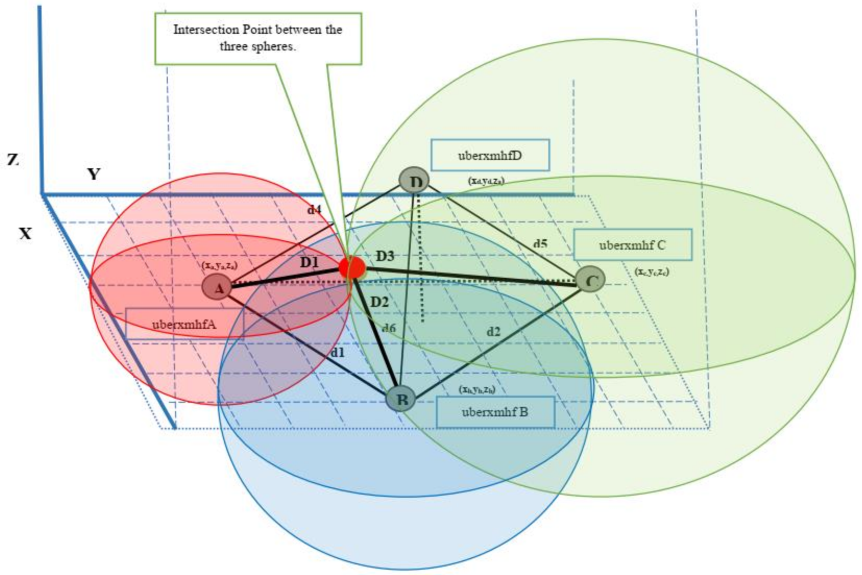


Figure 53. Intersection point calculation

Intersection computation algorithm

1. Pick one of the equations (EQ2) and subtract it from the other two (EQ1, EQ3). That will make those other two equations into linear equations in the three unknowns.
2. Use them to find two of the variables (x, y) as linear expressions in the third (z). These two equations are those of a line in 3-space, which passes through the two points of intersection of the three spheres.

3. Then substitute these into the equation of any of the original spheres (EQ1). This will give you a quadratic equation in one variable, which you can solve to find the two roots.
4. These values will allow you to determine the corresponding values of the other two variables, giving you the coordinates of the two intersection points. Keep the positive intersection point.

6.2.2.b Direct Location Method

The TOA measurements between the object and the stations are multiplied by the known signal propagation speed in the media to yield range measurements. Thus, (x_i, y_i, z_i), i = 1, ... 4 is the known position of station i. From the set of three equations from the measurements of the distances of the unknown node and the bases nodes, we can compute three equations expressed in z and y that can be cancelled out using straight forward algebra to get an explicit expression for x independent of y and z. Thus, we have one equation for each coordinate x, y and z with known parameters.

6.2.2.c 3D-TDOA *CHAN Algorithm*

In a localization system, time difference of arrival technique is widely used to estimate the location of a mobile station. Chan’s method is another non-iterative solution of achieving optimum performance for arbitrarily placed sensors. Following the derivation of the formulae for bias and mean-square errors of TDOA estimation under rm, this paper moves on to the joint estimation of TDOA and time scale. It proposes an iterative search for the maximization of the cross-ambiguity function (CAF), which is also the maximum likelihood function for additive Gaussian bandlimited white noise disturbance. In addition, a quadratic Lagrange interpolator is also proposed to obtain the initial parameter values for the iterative search, which can increase the chance of converging to the global minimum solution. It is necessary to time scale a digital sequence by a noninteger in the maximization process. For an N-point sequence, this operation, which first interpolates the samples by sinc functions and then resamples, is in the order O(N/sup 2/). Noting that the magnitude of the sinc function decreases rapidly from its peak, this paper uses a fast approximation (FA) method that applies only five sinc coefficients for the interpolation, reducing the computation to O(N).

6.2.2.d Hybrid 3D TOA/TDOA

TDOA measurements when converted to distance results in hyperbolic equations. N number of TOA measurements will result in N-1 hyperbolic equations. Using this concept with the proper modifications we can convert four equations using TDOA of 4 base nodes to a simplified linear matrix equation A x = b. The implementation is presented in the next session.

6.2.3 Verification

After the execution of the algorithm that is described above, the unknown node has its own coordinates in the system X (x_x, y_x, z_x). Those coordinates are shared between all the base nodes of the framework. Now we can use any verification criteria we want to accept or deny the node access in the system.

6.2.4 Node Access

Finally, after the verification phase is executed, the node can either have no access in the system since it has not the verification criteria necessary or have access to the system data. In this case, a safe and secure environment is created for the guest node by the hypervisor, where it can have access to the data system and can interact and communicate with it without data loses and with secure protection from any attacks.

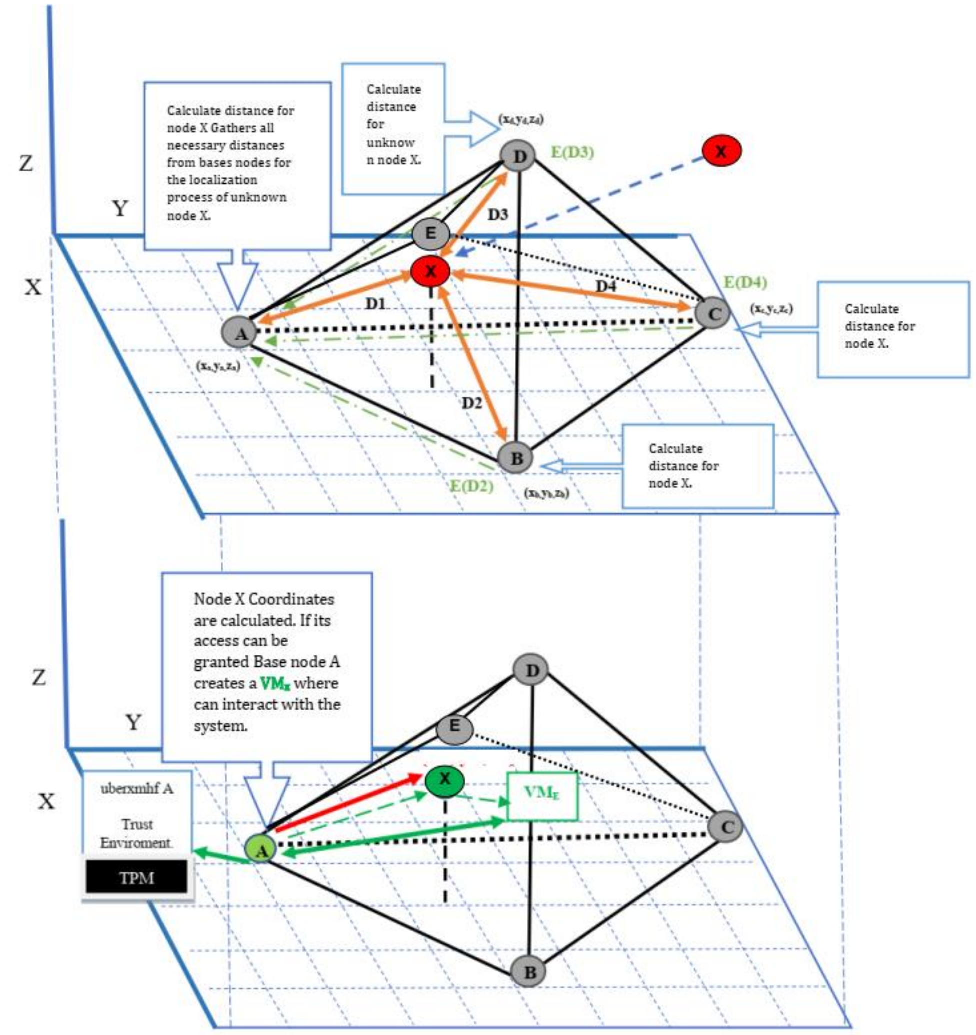

Calculate distance for node X Gathers all necessary distances from bases nodes for the localization process of unknown node X.
Calculate distance for unknow n node X.
(x_d,y_d,z_d)
D
E(D3)
X
Z
Y
X
E
D3
D1
D4
E(D4)
C
(x_c,y_c,z_c)
Calculate distance for node X.
A
(x_a,y_a,z_a)
D2
B
E(D2)
(x_b,y_b,z_b)
Calculate distance for node X.
Node X Coordinates are calculated. If its access can be granted Base node A creates a VMx where can interact with the system.
Z
Y
X
D
E
uberxmhf A
Trust Enviroment
TPM
X
VME
C
A
B

Figure 54. Verification and access overview.

6.3 Framework Implementation Overview

6.3.1 Simulation Overview

To evaluate the performance and implementation of our proposed framework, we simulated a random movement of the unknown node E in the field. The implementation of the movement algorithm is described in the next session, also the source code is provided in the appendixes. We assume that node X each time unit *t* has unknow coordinates:

- $X\,[\,x(t), y(t), z(t)]$

where $t \in \mathbb{N} \rightarrow t$ = [Start Movement, End Movement]. The random algorithm that we utilize to simulate this movement is Brownian motion. We can then utilize one localization method in order to calculate the distances of E from the bases nodes A, B, C and X and then calculate the coordinates *X [x(t), y(t), z(t)]* each time unit *t* with multi-lateration. The current distances of E from A, B, C, D is $D_N(t)$, N ∈ [A,B,C,D].

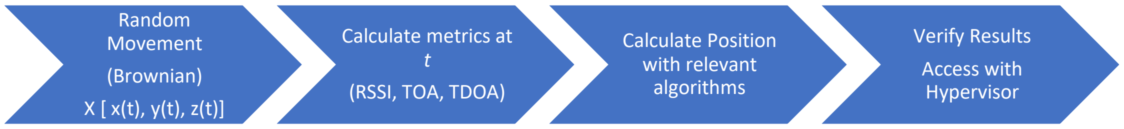


Figure 55. Simulation Movement steps

6.3.2 Random Movement

To calculate $D_N(t)$ we utilize and compared the performance of the localization techniques: RSSI, TDOA

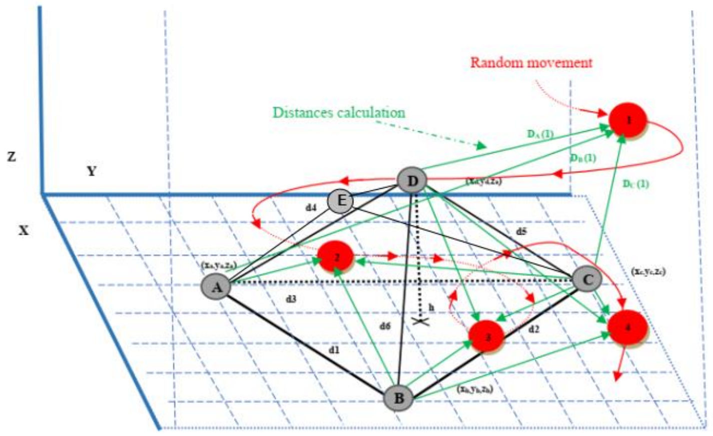


Figure 56. Unknown node movement

Brownian motion

Input in Brownian motion simulation: dimensions (m), the number of the walk step (n), the density of the field (d), a coefficient (t) and the start position $\hat{c}_0$.

m:	$m \in \mathbb{N}$ **&**	$0 < m < 4$
n:	$n \in \mathbb{N}$ **&**	$0 < n$
d:	$d \in \mathbb{R}$ **&**	$0 < d$
t:	$t \in \mathbb{R}$ **&**	$0 < t < 1$

$$\hat{c}_0: \quad \hat{c}_0 = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ \cdots \\ x_m \end{bmatrix} \quad \& \quad \hat{c_0} \in \mathbb{R}^{\mathrm{m}} \quad \& \quad x_{1\text{-}m} \in \mathbb{R}$$

Compute the Δt for utilize it as a new coefficient in the algorithm:

$$\Delta t = \frac{t}{n-1}$$

For every step n the algorithm will produce a new random m dimension position $\hat{c}_i = [\, x_1, x_2, \ldots\ldots, x_m]$

- ❖ $\forall\, i \in \{1, 2, \ldots\ldots, n\}$
  - ➢ S_i: $\quad S_i = \sqrt{2 * m * d * dt} * X_i$, where $X_i \sim U(\mathbb{R})$ **&** $S_i \in \mathbb{R}$
  - ➢ $\forall\, j \in \{1, 2, \ldots\ldots, m\}$
    - Δx^j_i : $\quad \Delta x^j_i = X^j_i$, where $X^j_i \sim U(\mathbb{R})$ **&** $\Delta x^j_i \in \mathbb{R}$
  - ➢ $\|x\|_i$: $\quad \|x\|_i = \sqrt{\sum_{k=1}^{k=m} (\Delta\chi_\iota^k)^2}$, where $\|x\|_i \in \mathbb{R}$
  - ➢ $\widehat{\mathrm{dx}_\iota}$: $\quad \widehat{\mathrm{dx}_\iota} = \begin{bmatrix} \Delta\chi_\iota^1 \\ \Delta\chi_\iota^2 \\ \cdots \\ \cdots \\ \Delta\chi_\iota^m \end{bmatrix}$, where $\widehat{\mathrm{dx}_\iota} \in \mathbb{R}^m$
  - ➢ $\widehat{\mathrm{dx}'_\iota}$: $\quad \widehat{\mathrm{dx}'_\iota} = \widehat{\mathrm{dx}_\iota} * \frac{s_i}{\|x\|_i} \rightarrow \widehat{\mathrm{dx}'_\iota} = \begin{bmatrix} \Delta\chi_\iota^1 * \frac{s_i}{\|x\|_i} \\ \Delta\chi_\iota^2 * \frac{s_i}{\|x\|_i} \\ \cdots \\ \cdots \\ \Delta\chi_\iota^m * \frac{s_i}{\|x\|_i} \end{bmatrix}$, where $\widehat{\mathrm{dx}'_\iota} \in \mathbb{R}^m$
  - ➢ $\hat{c}_i$: $\quad \hat{c}_i = \hat{c}_{i-1} + \widehat{\mathrm{dx}'_\iota}$, where $\hat{c}_i \in \mathbb{R}^m$

A new random m dimension position is produced $\hat{c}_i$. It will produce n random position in total. This algorithm simulates a m-D random movement of a node in n steps.

6.3.3 Calculate Metrics

6.3.3.a RSSI

As it was presented in chapter 3 [48], the formula for calculating the distances between two nodes using received signal strength indicator is:

$$P_R = P_T.\frac{G_T * G_R * \lambda^2}{(4*\pi)^2 * d^n} \ (1) \Rightarrow$$

$$P[\mathrm{dBm}] = 10 \cdot \log_{10} (P[W] \cdot 10^3) \ (2) \Rightarrow$$

$$RSSI = -\ (10 \cdot n \cdot \log_{10} d - A) \ (3) \Rightarrow$$

$$d = 10^{\frac{A-RSSI}{10*n}} \ (4)$$

Choosing value for n it depends on the environment. Typical values are: 2 for free space, 2.7 to 3.5 for urban areas, 3.0 to 5.0 in suburban areas and 1.6 to 1.8 for indoors when there is line of sight to the router. At maximum Broadcasting Power (+4 dBm) the RSSI ranges from -26 (a few inches) to -100 (40-50 m distance). Default transmit power for DD-WRT based routers is 70mW or 18.5dBm [49]

$$D_i = 10^{\frac{A_i - RSSI_i}{10*n}} \text{, where } i = \{1,2,3,4,5\}$$

6.3.3.b TOA

As it was presented in chapter 3, the formula for calculating the distances between two nodes using Time of Arrival is:

$$\Delta d_{i\text{-}x} = c * (\Delta t_{i\text{-}x})$$

Where c is the speed of light in the vacuum and Δt is the time difference between the start time and the time of arrival between a base node and unknown node X.

6.3.3.c TDOA

As it was presented in chapter 3, the formula for calculating the distances between two nodes using Time Difference of Arrival is:

$$\Delta d_{i\text{-}j\text{-}x} = c * (\Delta t_{i\text{-}j\text{-}x})$$

Where c is the speed of light in the vacuum and Δt is the time difference between the start time and the time of arrival between two base nodes and unknown node X. For example, if we have two base nodes i and j and x is the unknow node then:

$$\Delta d_{i\text{-}j\text{-}x} = c * (\Delta d_{i\text{-}x} - \Delta d_{j\text{-}x})$$

6.3.4 Calculate Position

6.3.4.a With Spheres Intersection points for RSSI/TOA Multilateration

Input: $x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3, r_1, r_2, r_3$. Subtract EQ2 from EQ1, move all constants to right side. Call the right side constant k1 [50]:

$k1 = r_1^2 - r_2^2 - x_1^2 + x_2^2 - y_1^2 + y_2^2 - z_1^2 + z_2^2$

Left side of EQ1 is of the form a1x + b1y + c1z where a1, b1, and c1 are the coefficients

$$a1 = 2 * (x_2 - x_1)$$

$$b1 = 2 * (y_2 - y_1)$$

$$c1 = 2 * (z_2 - z_1)$$

Subtract EQ2 from EQ3, move all constants to right side, Call the right side k3

$$k_3 = r_3^2 - r_2^2 - x_3^2 + x_2^2 - y_3^2 + y_2^2 - z_3^2 + z_2^2$$

Left side of EQ3 is of the form a3x + b3y + c3z, where a3, b3, and c3 are the coefficients

$$a3 = 2 * (x_2 - x_3)$$

$$b3 = 2 * (y_2 - y_3)$$

$$c3 = 2 * (z_2 - z_3)$$

The two equations (EQ1, EQ3) are now linear equations in the three unknowns: EQ1: a1x + b1y + c1z = k1, EQ3: a3x + b3y + c3z = k3. Then find y as a linear expression of z. y = e*z + f

$$\text{IF :}\begin{cases} \text{a1} = 0 \quad => \begin{cases} e = \frac{-c1}{b1} \\ f = \frac{-c3}{b3} \end{cases} \\ \text{a1} \neq 0 \text{ AND a3} = 0 => \begin{cases} e = \frac{-c3}{b3} \\ f = \frac{k3}{b3} \end{cases} \\ \text{a1} \neq 0 \text{ AND a3} \neq 0 => \begin{cases} a31 = \frac{k3}{b3} \\ e = -\frac{\text{a31} * \text{c1} - \text{c3}}{\text{a31} * \text{b1} - \text{b3}} \\ f = \frac{\text{a31} * \text{k1} - \text{k3}}{\text{a31} * \text{b1} - \text{b3}} \end{cases} \end{cases}$$

Then find x as a linear expression of z. x = g*z + h ➔

$$\text{IF :}\begin{cases} \text{b1} = 0 \quad => \begin{cases} g = \frac{-c1}{a1} \\ h = \frac{-c3}{a3} \end{cases} \\ \text{b1} \neq 0 \text{ AND b3} = 0 => \begin{cases} g = \frac{-c3}{a3} \\ h = \frac{k3}{a3} \end{cases} \\ \text{b1} \neq 0 \text{ AND b3} \neq 0 => \begin{cases} b31 = \frac{b3}{a3} \\ g = -\frac{\text{b31} * \text{c1} - \text{c3}}{\text{b31} * \text{a1} - \text{a3}} \\ h = \frac{\text{b31} * \text{k1} - \text{k3}}{\text{b31} * \text{a1} - \text{a3}} \end{cases} \end{cases}$$

Substitute these into the equation of any of the original spheres (EQ1). This will give you a quadratic equation in one variable, which you can solve to find the two roots

$$A = g^2 + e^2 + 1$$

$$B = -x_1 * g - y_1 * e - 2 * z_1 - x_1 * g - y_1 * e + 2 * g * h + 2 * e * f$$

$$C = x_1^2 + y_1^2 + z_1^2 - 2 * x_1 * h - 2 * y_1 * f + h^2 + f^2 - r_1^2$$

Use the quadratic formula to solve to find the two roots.

$$\text{rootD} = \sqrt{(B^2 - 4 * A * C)}$$

$$z = \frac{-B + \text{rootD}}{2 * A},\; z\_ = \frac{-B - \text{rootD}}{2 * A}$$

Calculate you the coordinates of the two intersection points.

$$x = g * z + h, \quad x\_ = g * z\_ + h$$

$$y = e * z + f, \quad y\_ = e * z\_ + f$$

Finally, keep the positive solution:

$$\text{E: Solutions} => \begin{cases} (x, y, z) \\ (\_x, \_y, \_z) \end{cases} \xRightarrow{\text{positive result}} \begin{cases} (x, y, z) \\ \cancel{(\_x, \_y, \_z)} \end{cases} => \textbf{Solution: E = (x, y, z)}$$

6.3.4.b Direct Location Method for RSSI/TOA

The TOA measurement equation is written as

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = r_i^2, i = 1,2,3\ 4$$

Substituting i = 2, 3, 4 in Eq. above and subtracting each of the three equations from Eq. above when i = 1, we get three equations of the form:

$$A_j, i = 2, 3, 4\ A_j = (r_1 - r_i) + (x_i - x_1) + (y_i - y_1) + (z_i - z_1), j = 1, 2, 3$$

Let

$I_1 = (z_3 - z_1)*(x_2 - x_1)-(z_2 - z_1)*(x_3 - x_1)$, $I_2 = (z_4 - z_1)*(x_2 - x_1)-(z_2 - z_1)*(x_4 - x_1)$

$I_3 = (z_3 - z_1)*A_1 - (z_2 - z_1) * A_2$, $I_4 = (z_4 - z_1)* A_1 - (z_2 - z_1) * A_3$

$I_5 = (z_3 - z_1) * (y_2 - y_1) - (z_2 - z_1) * (y_3 - y_1)$, $I_6 = (z_4 - z_1)*(y_2 - y_1)-(z_2- z_1)*(y_4 - y_1)$

$I_7 = (y_3 - y_1) *(x_2 - x_1) - (y_2 - y_1)*(x_3 - x_1)$, $I_8 = (y_4 - y_1)*(x_2 - x_1) - (y_2 - y_1)*(x_4 - x_1)$

$I_9 = (y_3 - y_1) * A_1 - (y_2 - y_1) * A_2$, $I_{10} = (y_4 - y_1)* A_1 - (y_2 - y_1)* A_3$

$I_{11} = (y_3 - y_1)*(z_2 - z_1) - (y_2 - y_1)*(z_3 - z_1)$, $I_{12} = (y_4 - y_1)*(z_2 - z_1) - (y_2 - y_1)*(z_4 - z_1)$

Thus, the solution can be calculated as presented below:

$$\mathbf{x}= \frac{I_4*I_5 - I_3*I_6}{I_2*I_5 - I_1*I_6} * \frac{1}{2}, \mathbf{y}= \frac{I_4*I_5 - I_3*I_6}{I_2*I_5 - I_1*I_6} * \frac{1}{2}, \mathbf{z}= \frac{I_4*I_5 - I_3*I_6}{I_2*I_5 - I_1*I_6} * \frac{1}{2}$$

6.3.4.c With TDOA CHAN

It is assumption that M BSs are randomly distributed in 3-dimension space. (x, y, z) is unknown MS position and (X_i, Y_i, Z_i) is BS_i position. R_i is the distance between BS_i and MS. Thus, the distance between MS and BSi is given by:

$$R_i = (X_i - x)^2 + (Y_i - y)^2 + (Z_i - z)^2 = K_i - 2*X_{i*}x - 2*Y_{i*}y - 2*Z_{i*}z + R = (c * t_i)^2, \text{ where } i=1, 2.., M$$

Where $K_i = X_i^2 + Y_i^2 + Z_i^2$, $R = x^2+y^2+z^2$, τ_i the value of TOA, c is the speed of light. Assume $z_a = [z_p^T, R]^T$ is unknown vector, where $z = [x, y, z]^T$ .The error vector with TOA noise is that: $\psi = h - G_a z_a^0$ where,

$$h = \begin{bmatrix} R_1^2 - K_1 \\ \dots\dots\dots\dots \\ R_M^2 - K_M \end{bmatrix}, G_a \begin{bmatrix} -2X_M - 2Y_M - 2Z_M\ 1 \\ \dots\dots\dots\dots \\ -2X_M - 2Y_M - 2Z_M\ 1 \end{bmatrix}$$

By using WLS method, we use covariance matrix Q of measured value of TOA to replace the Covariance matrix of error vector ψ. za is given by:

$$z_a = \operatorname{argmin}\{(h-G_a z_a)^T Q^{-1}(h-G_a z_a)\} = (G_a^T Q^{-1} G_a)^{-1}(G_a^T Q^{-1} h), \text{ where } Q = \mathit{diag}(\sigma_1^2, \dots, \sigma_M^2)$$

If error of TOA is little, ψ is given by: $\psi = 2Bn + n*n \approx 2Bn$, where $B = \mathit{diag}(R_1^2, \dots, R_M^2)$, R_i^0 is the actual distance between MS and BS_i, n is the measurement error of TOA. Covariance matrix of error vector ψ is given by: $\psi = E[\psi\psi^T] = 4BQB$. In the environment with noise,

$$R_i = R_i^0 + cn_i,\ G_a = G_a^0 + \Delta G_a,\ h = h^0 + \Delta h$$

It is assumed that $z_a = z_a^0 + \Delta z_a$, Δz_a and covariance matrix of Δz_a are measured by:

$$\Delta z_a = c * (G_a^T \psi G_a)^{-1} G_a \psi^{-1} Bn,\ \operatorname{cov}(z_a) = E[\Delta z_a\, \Delta z_a^T] = (G_a^T \psi^{-1} G_a)^{-1}$$

z_a is a zero-mean random variable, thus z_a can be expressed as:

$$z_{a,1} = x^0 + e_1,\ z_{a,2} = y^0 + e_2,\ z_{a,3} = z^0 + e_3,\ z_{a,4} = R^0 + e_4$$

Where e_1, e_2, e_3, e_4 are the error estimates of z_a. Vector's error of z a can be expressed as: $\psi' = h' - G_a' z_a'$

$$h' = \begin{bmatrix} z_{a,1}^2 \\ z_{a,2}^2 \\ z_{a,3}^2 \\ z_{a,4}^2 \end{bmatrix}, G_a' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, z_a' = \begin{bmatrix} x^2 \\ y^2 \\ z^2 \end{bmatrix}$$

The covariance matrix of ψ ' also could be defined by: $\psi' = E[\psi'\psi'^T] = 4B'\operatorname{cov}(z_a)B'$. Where $B = \mathit{diag}(x_0, y_0, z_0, 1/2)$, x^0, y^0, z^0 in B' could be replaced by the value of z_a. The estimates of z'_a by WLS is given by: $z'_a = (G_a' \psi'^{-1} G'_a)^{-1}(G'^T_a \psi'^{-1} h')$. MS positioning is calculated by:

$$\mathbf{z_p = \sqrt{z_a'} \text{ or } z_p = -\sqrt{z_a'}}$$

The plus or minus selection of (x,y,z) in z_p should be the same as the sign of (x,y,z) in z_a .

6.3.4.d With Hybrid TOA/TDOA

The equation for the TOA measurement of the signal at the i-th antenna station transmitted from an emitter located at x = (x, y, z) is:

$$\tau_i = \frac{\sqrt{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2}}{c} \Rightarrow (\tau_i * c)^2 = (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 \Rightarrow$$

$$(\tau_i * c)^2 = (x^2+y^2+z^2) - (2xx_i+2yy_i+2zz_i) + (x_i^2+ y_i^2+ z_i^2)$$

Let

$$R = (x^2+y^2+z^2) \text{ and } K_i = (x_i^2+ y_i^2+ z_i^2) \Rightarrow$$

$$R_i^2 = R - (2xx_i+2yy_i+2zz_i) + K_i$$

TDOA measurement between the i-th and the j-th antenna station pair is obtained as:

$$\tau_{ij} = \tau_i - \tau_j = \frac{\sqrt{(x-x_i)^2+ (y-y_i)^2+ (z-z_i)^2}}{c} - \frac{\sqrt{(x-x_i)^2+ (y-y_i)^2+ (z-z_i)^2}}{c}$$

We calculate the values τ_{13}, τ_{14}, τ_{23}, τ_{14} we get to []:

$$a_{134} = x \times b_{134} + y \times c_{134} + z \times d_{134}$$

$$a_{234} = x \times b_{234} + y \times c_{234} + z \times d_{234}$$

Finally, we get a matrix form Ax = b

$$A = \begin{bmatrix} 2(x_1 - x_2)2(x_3 - x_4) & 2(y_1 - y_2)2(y_3 - y_4) & 2(z_1 - z_2)2(z_3 - z_4) \\ \frac{x_{31}}{R_{13}} - \frac{x_{41}}{R_{14}} & \frac{y_{31}}{R_{13}} - \frac{y_{41}}{R_{14}} & \frac{z_{31}}{R_{13}} - \frac{z_{41}}{z_{14}} \\ \frac{x_{32}}{R_{23}} - \frac{x_{42}}{R_{24}} & \frac{y_{32}}{R_{23}} - \frac{y_{42}}{R_{24}} & \frac{x_{31}}{R_{13}} - \frac{x_{31}}{R_{13}} \end{bmatrix}$$

$$b = \begin{bmatrix} (R_1^2 - R_2^2 - K_1 + K_2)(R_3^2 - R_4^2 - K_3 + K_4) \\ 0.5(R_1^2 - R_2^2 + \frac{K_{14}}{R_{14}} - \frac{K_{13}}{R_{13}}) \\ 0.5(R_1^2 - R_2^2 + \frac{K_{24}}{R_{24}} - \frac{K_{23}}{R_{23}}) \end{bmatrix}$$

where we solve for $\mathbf{x} = [\, x, y, z]^T$

7. PROPOSED FRAMEWORK SIMULATION

In this chapter we present our proposed Localization framework simulation. We explain how we installed the relevant components and features necessary for our framework alongside with our implemented code and scripts. Also, we present how the simulation works and how we tested it.

7.1 Framework simulation overview

The simulator can be found in [64]. The simulator contains two basic phases. In the first is the process of calculating all the metrics values (TDOA, TOA, RSSI) of the base nodes from the unknown and then simulate the algorithms performance. In this phase we implemented a python project that is responsible for all the above. In the second phase we implemented a bash script that is responsible for downloading and installing the hypervisor to a local machine alongside with all the necessary modules and packets. We explain with details those implementations in the sections below.

7.1.1 Localization algorithms simulation process

The python simulator contains of 6 phases:

1. **Browian Motion**
   In this phase we create a random motion of unknown node X in a 3d field
2. **Data creation**
   Here we create the RSSI, TOA and TDOA values for each position of X.
3. **Simulation**
   This is the phase where the algorithms are executed.
4. **Storing**
   We store the data to files.
5. **Graphics**
   Here we plot all the movements and algorithms traces
6. **Statistics**
   Finally, we evaluate the algorithm’s performances.

More information for the implementation of the script are presented below.

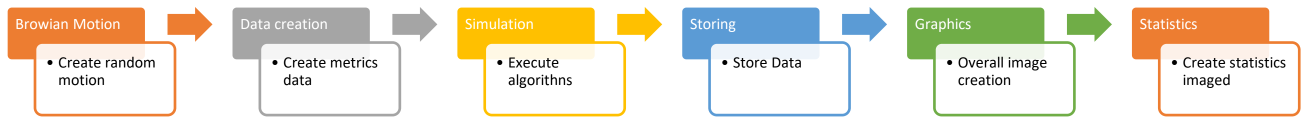


Figure 57. Geolocation simulation process

7.1.2 Hypervisor simulation process

The bash simulator contains of 6 phases:

1. **Update**
   Update and upgrade software of local machine.
2. **Download**
   Download hypervisor from Github.
3. **Installation**
   Install hypervisor to local environment.
4. **Configuration**
   Install all relevant packet and features alongside with any configuration.
5. **Verification**
   Verify that hypervisor is installed properly.

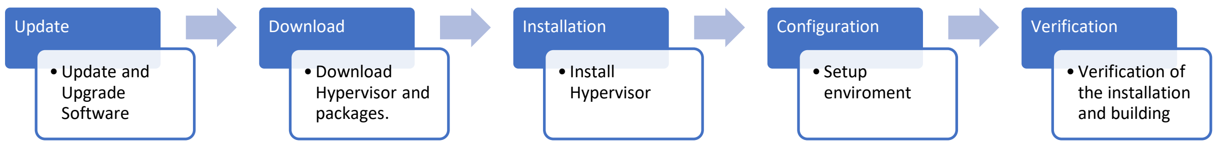

Update
• Update and Upgrade Software
Download
• Download Hypervisor and packages.
Installation
• Install Hypervisor
Configuration
• Setup enviroment
Verification
• Verification of the installation and building

Figure 58. Hypervisor simulation process

7.1.3 Framework Folder Structure

Here we present a short overview of the folder structure in order to understand how the files and folders are placed. The brown scaled colored boxes represents folders and the blue scaled boxes files. The green is the executable python file to start the simulation. The input is given from the JSON file in the folder Input, grey box.

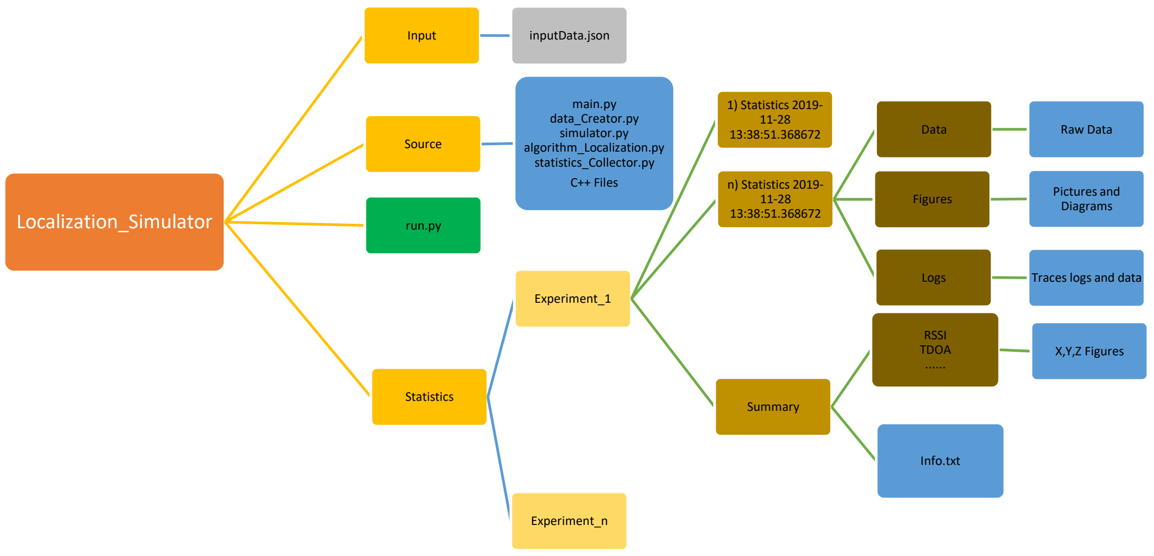


Figure 59. Simulation folder structure

7.1.4 Framework Implementation

Our simulator takes as an input a JSON file. This JSON file contains the information that are present below:

Table 10. JSON input data simulation

Input	Explanation	Example
X	Start position of unknown node	"X": [20, -20, 20],
nodes	Base nodes coordinates	"A": [10, -40, 0]
algorithms	Algorithm to execute	"RSSI": ["Multilateration"],
transittionPower_dBm	Base node transmission power	"transittionPower_dBm": 70,
noise	Noise coefficient 0 – 5	"noise": 3.5,
Dimensions	Dimensions, 3D	"Dimensions": 3,
steps	Steps of unknow node	"steps": 500,
density	3D field density	"density": 1000.0
time_Coeff	Time coefficient 0-1.0	"time_Coeff": 1.0
totalExperiments	Samples per experiment	"totalExperiments": 1
showPlots	Show plot for each sample	"showPlots": "Yes"
PickForCalculation	Select pick base node for calculation	"PickForCalculation": "Yes"

In the JSON file we can define how many experiments we would like to execute alongside with how many samples should we consider for each, for a general statistic evaluation. Each input in the JSON file represent an individual experiment. For each experiment, a number of samples, that are defined from the input, are executed. Those sample are individual tests that contains the python simulation process that was described above in 7.1.1. Then the python simulator generates each experiment with the relevant sample and for each experiment provides the overall statistics evaluation that is calculated from its sample. All the necessary information, log files, data that is utilized, figures etc. are provided and generated in the relevant folders.

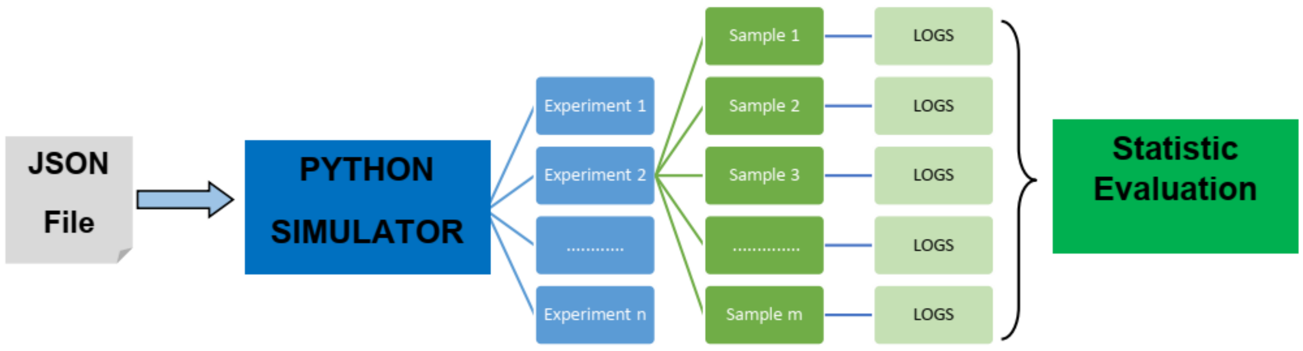


Figure 60. Python Simulator overview

Base Stations communication process

In order to compute and evaluate the time performance of the simulator we developed a C++ application to simulate the environment. The simulation contains the unknown node X where broadcasts its presents by sending messages in the environment, base station A, B and C where receives the messages from X and they forward the data to the Master base station where gathers all the receives metrics from X and computes its position. The communication time is computed from the first presence of node X until the last step of the simulation.

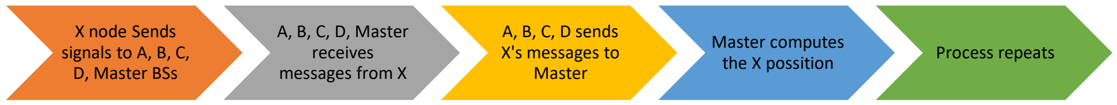
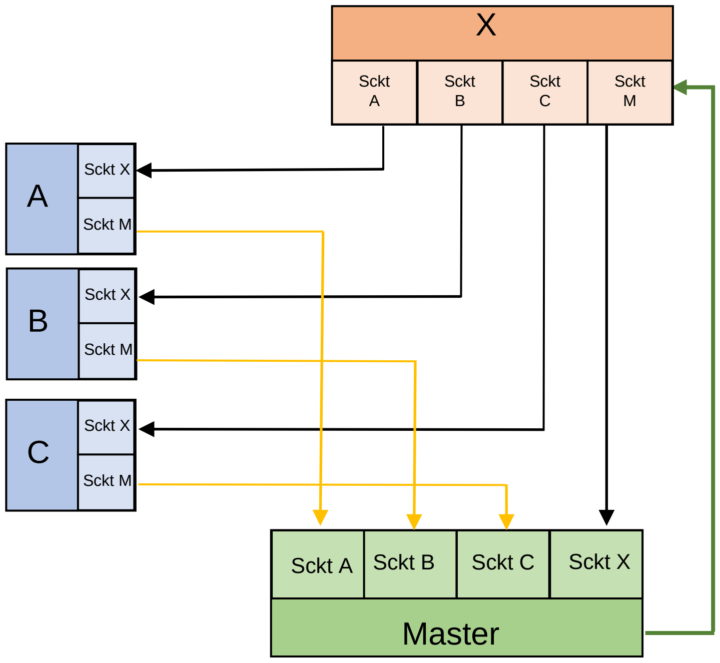

X node Sends signals to A, B, C, D, Master BSs
A, B, C, D, Master receives messages from X
A, B, C, D sends X's messages to Master
Master computes the X possition
Process repeats

Figure 61. Base stations communication process

To simulate this process, we utilized the process/sockets concept. Each base station A, B and C, the master station and unknown X are individual C++ processes. Each base station contains of two sockets one to receive messages from X and one to send to master. Node X contains four sockets for all the stations each to send messages. The master contains three sockets for the base stations and one for node X to receive messages. The simulation starts from X which continuously send messages to all BSs and stops when the termination message is send. The overall communication time is computed.



X sends messages which corresponds to simulated metrics such as RSSI signal, to A, B, C, M. Each BSs A, B and when receives from X immediately notify the Master. When the master receives four messages then can compute the position of X. To end the simulation node X sends and “END” message and the simulation stops.

Figure 62. Base communication architecture

7.2 Framework Requirements

In our implementation we use python 3.7 for the creation of a python project for the localization algorithms evaluation. We have implemented the hypervisor's execution in a raspberry environment. Below we present the requirements in order to execute the simulation.

7.2.1 Localization simulation

1. **OS: Ubuntu 16.04**
2. **Python 3.7, Pip 3**
3. **Python Packages** Math, matplotlib, numpy, random, json, scipy, datetime, mpl_toolkits, urllib3

7.2.2 Hypervisor simulation

In our implementation, our hardware and OS are presented below:

1. **OS: A Raspberry PI v3 board.** That ensure that the model B or B+ with the Cortex-A53 quad-core processor
    i. Hardware Virtualization extensions: Cortex A53: Hardware Support for Virtualization
    ii. 2nd-level page tables Typically turned on implicitly along with Virtualization extensions Cortex A53: Second-stage Page Tables
2. **TPM: version 2.0**

7.3 Framework Execution

In order to start the simulation after the input is provided in Input/inputData.json, the following command must be executed from command line:

```
~$ python3.7 run.py
```

The source code run.py can also be executed with python2. It will start the evaluation of the packets and features that are installed in the system. If all the necessary packets are installed, then it will evaluate the input in /Input/. If the input contains plausible and valid data, then the simulation will start.

All the necessary folders are created, and the simulator is responsible for creating graphs, pictures and log files for better understanding. Each experiment created is unique since a time stamp is utilized with the current time to avoid conflicts.

7.4 Simulation input data

7.4.1 Field input

Three dimensions field m = 3, Field density d = 1000

7.4.2 Nodes coordinates input

Table 11. Nodes Coordinates input

Nodes	x-Axis	y-Axis	z-Axis		Distances Between Nodes			
				A	B	C	D	E
A	10	-40	0	-	83.2165	80.7774	57.4456	56.5685
B	-35	30	0	83.2165	-	75.1664	61.0327	90.1387
C	40	35	0	80.7774	75.1664	-	66.5206	36.4005
D	0	0	40	57.4456	61.0327	66.5206	-	64.0312
E	50	0	0	56.5685	90.1387	36.4005	64.0312	-
X Start (C_0)	0	-20	20	30.0000	64.2261	70.8872	28.2842	57.4456

7.4.3 Brownian motion input

Three dimensions m = 3, Field density d = 1000, Number of random walk steps n = 500, Coefficient t = 0.1.

7.4.4 Extra parameters input

Transmitted power of Base Nodes, 70mW or 18.5dBm, n = 3, average noise coefficient value, tmD = 3, average transmission time delay value in seconds

7.5 Simulation output data

7.5.1 Graphics

Plotting the graphics of TDOA, TOA and RSSI calculated values from base nodes.

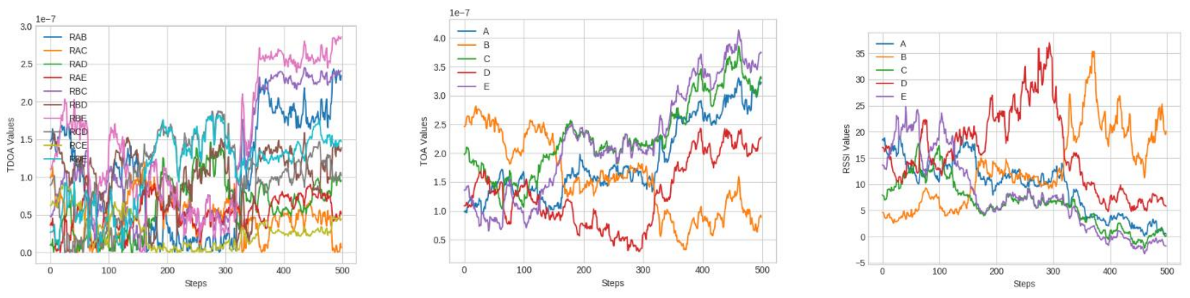


Figure 63. TDOA, TOA and RSSI simulated values

Plotting the graphics of all the algorithm x, y, z values compared to the real values.

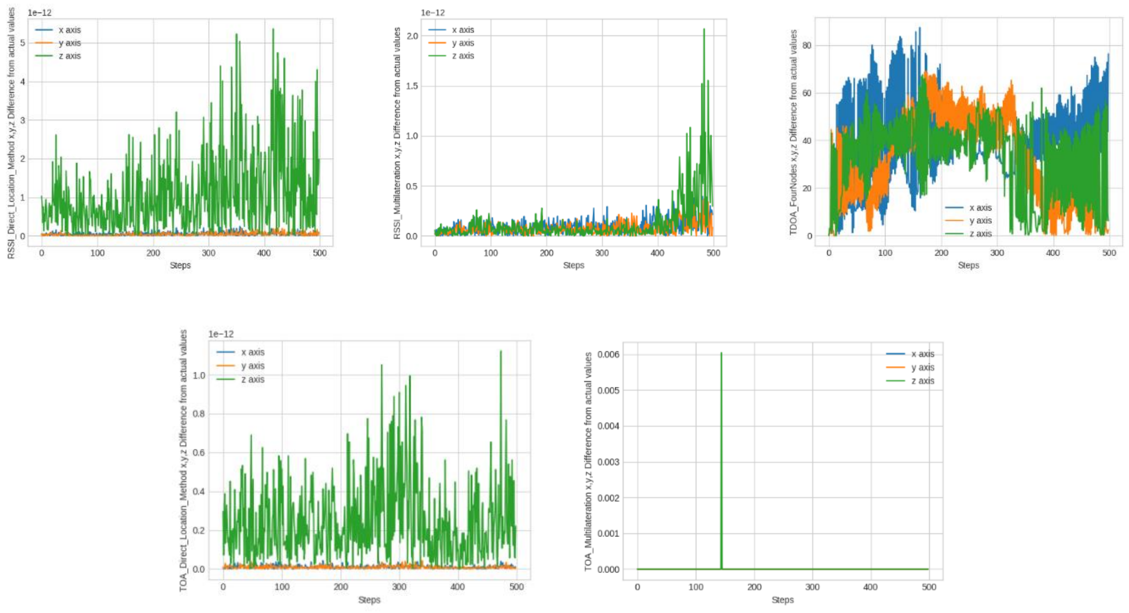


Figure 64. X, Y, Z performance from algorithms

Plotting the graphics of all the algorithm time of execution and communication time.

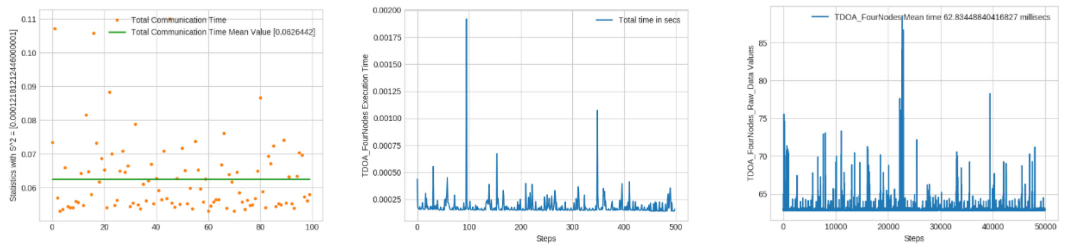


Figure 65. time performance algorithm

Plotting the motion graph of the unknown node alongside with the calculated traces.

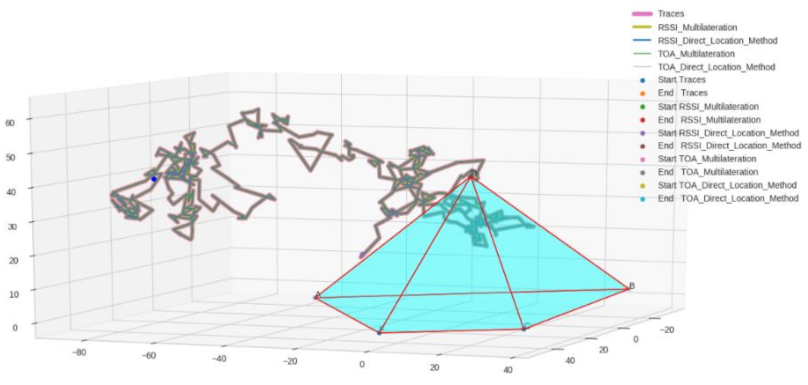


Figure 66. Unknown node motion

7.5.2 Statistics files

For each statistic file contains three folders: The Data , where raw data is provided for the time measurements , the metrics values and the traces, the Figures where the relevant diagrams and figure are provided and the Logs folder where the output logs of the simulator is provided.

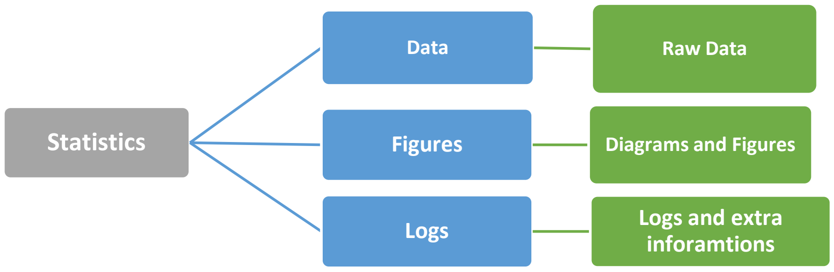


Figure 67. Statistics files structure

7.5.3 Summary files

In the summary folder a summary is computed from all the statistics samples that are performed in each experiment. It contained to main parts. The Performance, where it is evaluated for each algorithm the computation performance (the computed X, Y, Z values with the real X, Y, Z values) for the error evaluation and the Time performance where the total execution and communication time is summarized for each algorithm. In each part relevant figures and diagrams are provided alongside with a text file with general info. The mean value and standard deviation is provided for each.

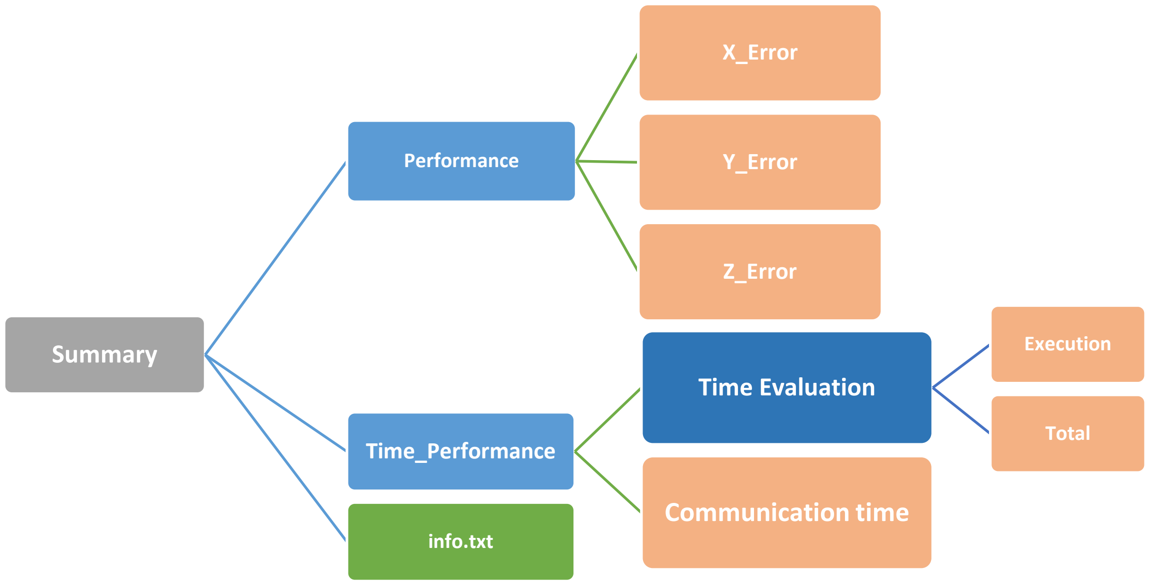


Figure 68. Summary files structure

8. VALIDATION

For each algorithm we executed one experiment with 100 samples each. Each sample contained 500 steps of the unknow node which they correspond to individual evaluation of each algorithm. We created a statistical evaluation for each algorithm, where we compered the localization error of the calculated coordinates from the real values.

8.1 Localization algorithms accuracy performance validation

We performed the experiments to the environment below for each:

1. PC-Intel_Core_i3-1.7GHz_RAM4GB
2. PC-Intel_Core_i7-2.7GHz_RAM8GB
3. PC-Intel_Core_i7-2.7GHz_RAM16GB
4. Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB

8.1.1 3D-RSSI Multilateration

From a sample of 50000 steps overall, we computed the mean error value for each coordinate x, y, z alongside with the standard deviation for 3D-RSSI Multilateration. We execute the simulation for four different environments as presented below.

8.1.1.a 3D-RSSI Multilateration PC-Intel_Core_i3-1.7GHz_RAM4GB

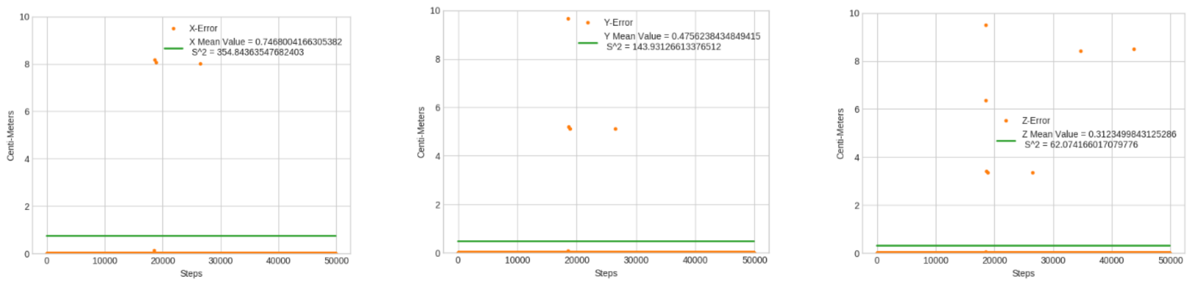


Figure 69. X, Y, Z RSSI Mult Alg, PC-Intel_Core_i3-1.7GHz_RAM4GB

8.1.1.b 3D-RSSI Multilateration PC-Intel_Core_i7-2.7GHz_RAM8GB

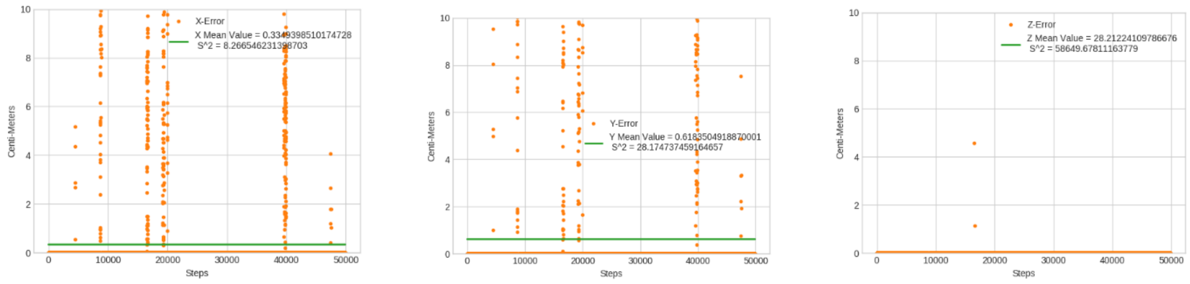


Figure 70. X, Y, Z RSSI Mult Alg, PC-Intel_Core_i7-2.7GHz_RAM8GB

8.1.1.c 3D-RSSI Multilateration PC-Intel_Core_i7-2.7GHz_RAM16GB

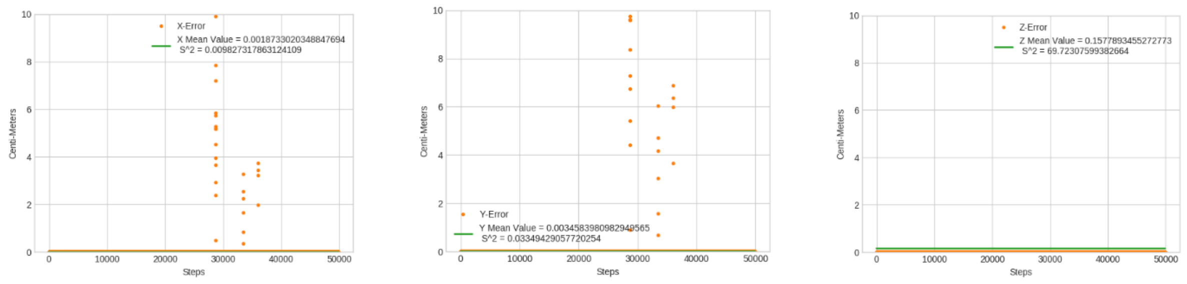


Figure 71. X, Y, Z RSSI Mult Alg, PC-Intel_Core_i7-2.7GHz_RAM16GB

8.1.1.d 3D-RSSI Multilateration Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB

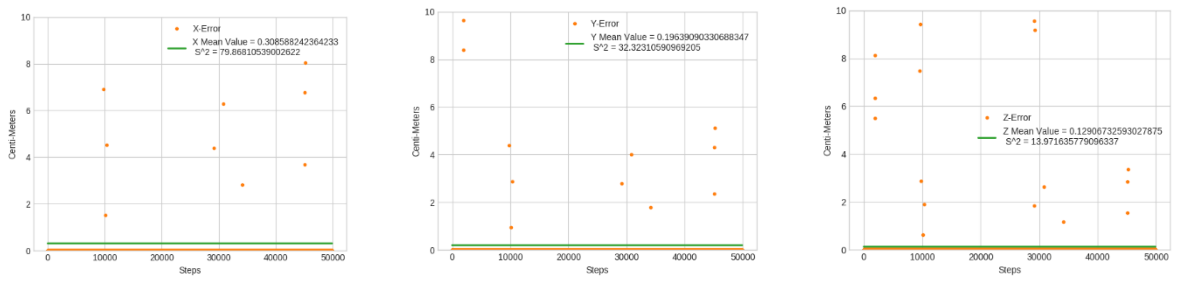


Figure 72. X, Y, Z RSSI Mult Alg, Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB

8.1.2 3D-TOA Multilateration

From a sample of 50000 steps overall, we computed the mean error value for each coordinate x, y, z alongside with the standard deviation for 3D-TOA Multilateration. We execute the simulation for four different environments as presented below.

8.1.2.a 3D-TOA Multilateration PC-Intel_Core_i3-1.7GHz_RAM4GB

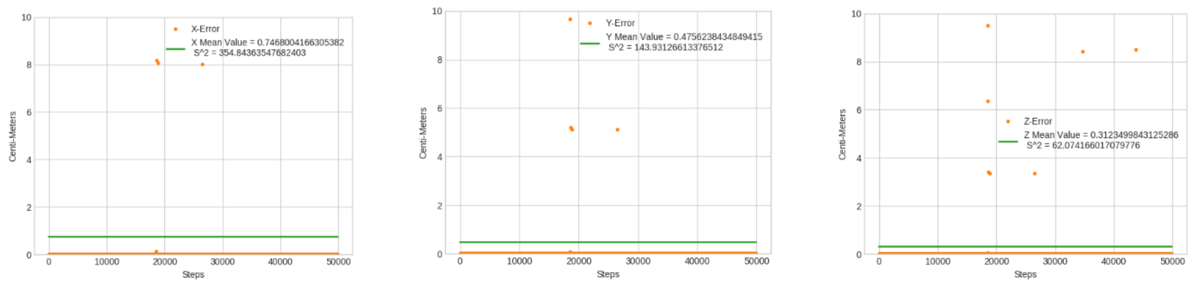


Figure 73. X, Y, Z TOA Mult Alg, PC-Intel_Core_i3-1.7GHz_RAM4GB

8.1.2.b 3D- TOA Multilateration PC-Intel_Core_i7-2.7GHz_RAM8GB

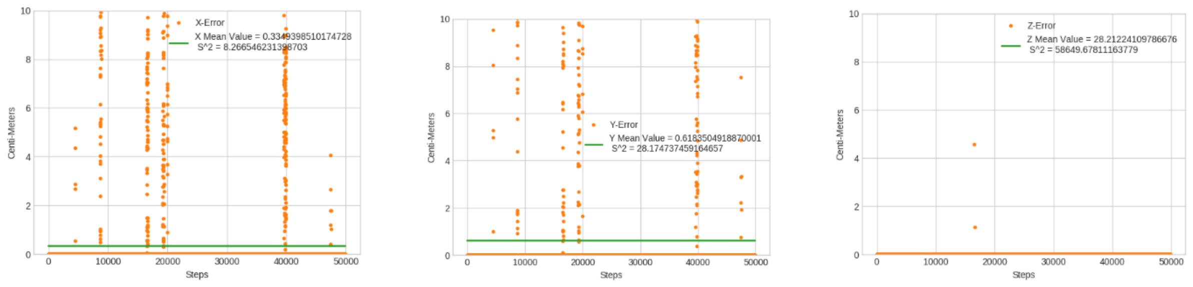


Figure 74. X, Y, Z TOA Mult Alg, PC-Intel_Core_i7-2.7GHz_RAM8GB

8.1.2.c 3D- TOA Multilateration PC-Intel_Core_i7-2.7GHz_RAM16GB

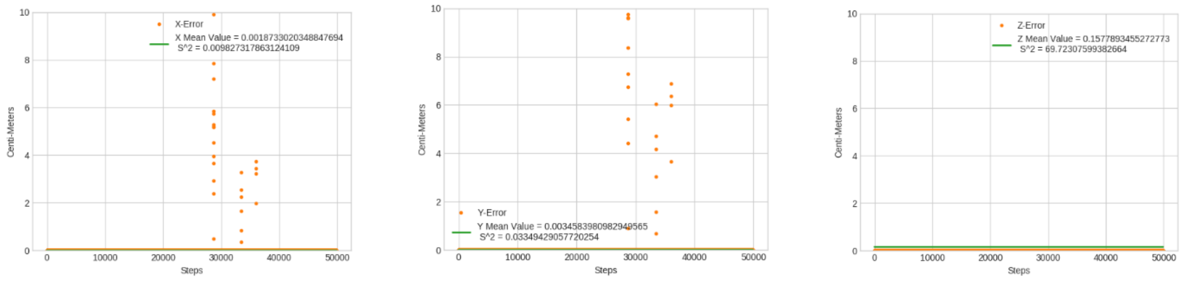


Figure 75. X, Y, Z TOA Mult Alg, PC-Intel_Core_i7-2.7GHz_RAM16GB

8.1.2.d 3D- TOA Multilateration Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB

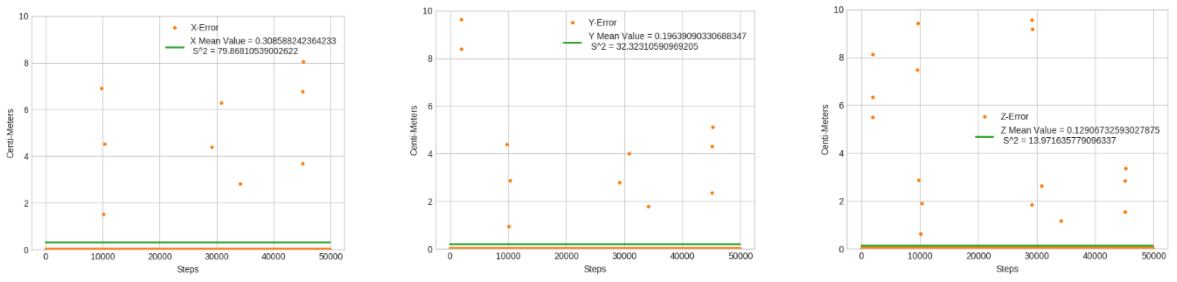


Figure 76. X, Y, Z TOA Mult Alg, Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB

8.1.3 3D-RSSI/TOA Direct method

From a sample of 50000 steps overall, we computed the mean error value for each coordinate x, y, z alongside with the standard deviation for 3D-RSSI Direct method. We execute the simulation for four different environments as presented below.

8.1.3.a 3D-RSSI Direct method PC-Intel_Core_i3-1.7GHz_RAM4GB

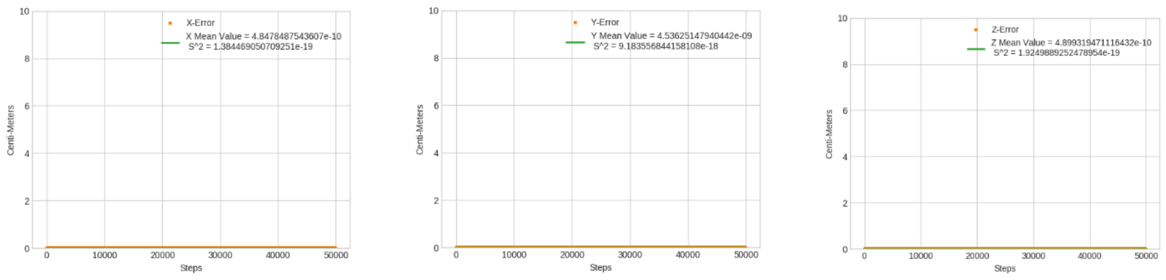


Figure 77. X, Y, Z RSSI Direct Alg, PC-Intel_Core_i3-1.7GHz_RAM4GB

8.1.3.b 3D-RSSI Direct method PC-Intel_Core_i7-2.7GHz_RAM8GB

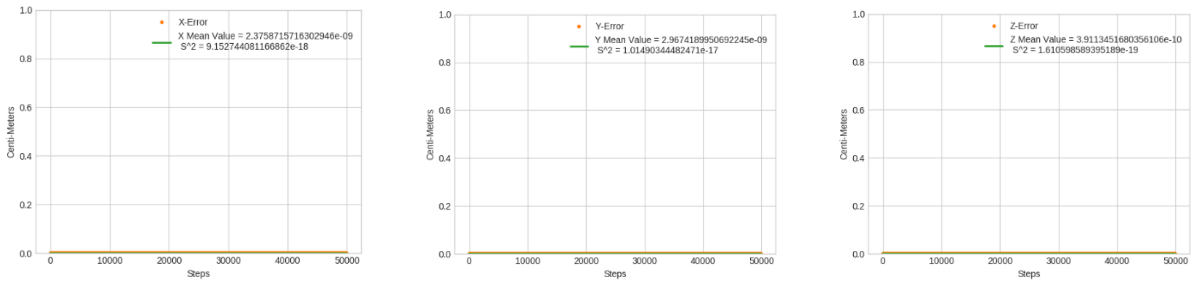


Figure 78. X, Y, Z RSSI Direct Alg, PC-Intel_Core_i7-2.7GHz_RAM8GB

8.1.3.c 3D-RSSI Direct method PC-Intel_Core_i7-2.7GHz_RAM16GB

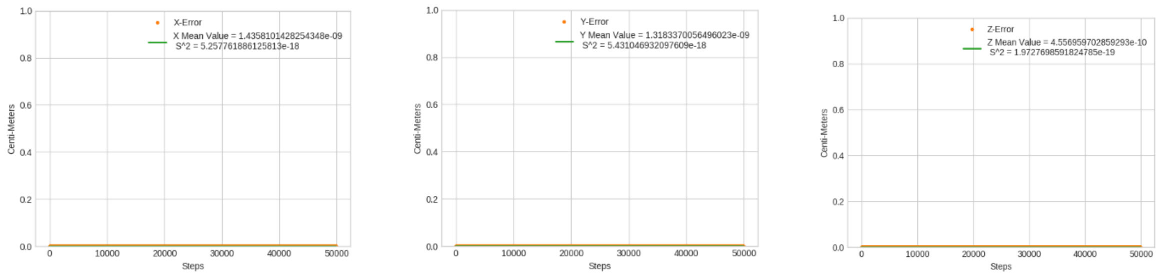


Figure 79. X, Y, Z RSSI Direct Alg, PC-Intel_Core_i7-2.7GHz_RAM16GB

8.1.3.d 3D-RSSI Direct method Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB

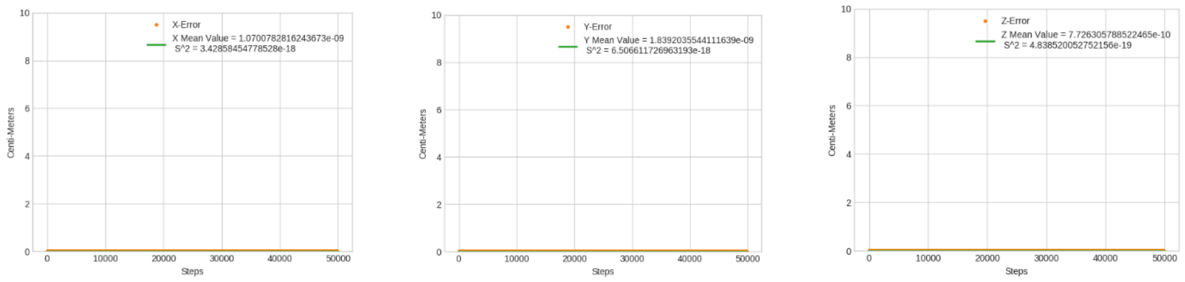


Figure 80. X, Y, Z RSSI Direct Alg, Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB

8.1.4 Hybrid 3D-TOA/TDOA

From a sample of 50000 steps overall, we computed the mean error value for each coordinate x, y, z alongside with the standard deviation for Hybrid 3D-TOA/TDOA. We execute the simulation for four different environments as presented below.

8.1.4.a Hybrid 3D-TOA/TDOA PC-Intel_Core_i3-1.7GHz_RAM4GB

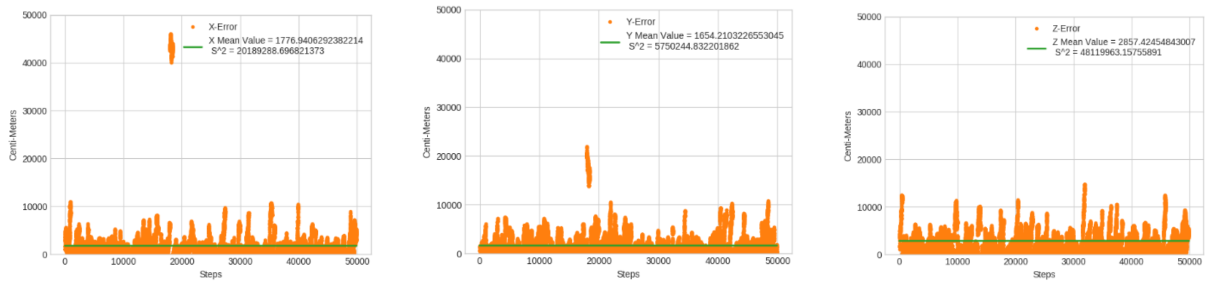


Figure 81. X, Y, Z Hybrid 3D-TOA/TDOA, PC-Intel_Core_i3-1.7GHz_RAM4GB

8.1.4.b Hybrid 3D-TOA/TDOA PC-Intel_Core_i7-2.7GHz_RAM8GB

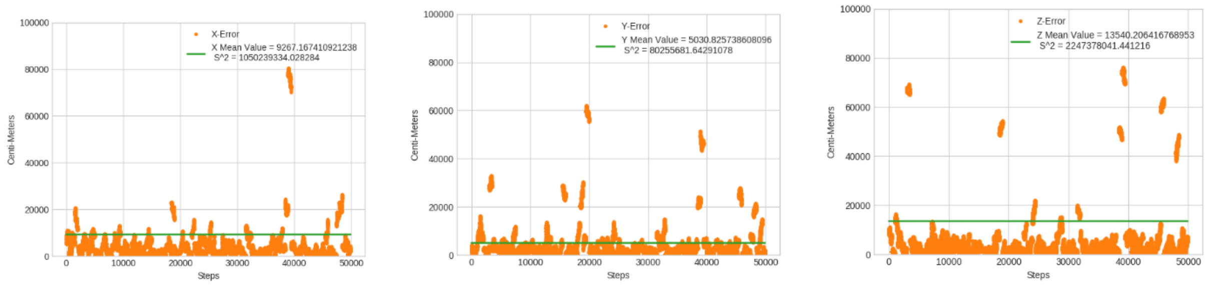


Figure 82. X, Y, Z Hybrid 3D-TOA/TDOA, PC-Intel_Core_i7-2.7GHz_RAM8GB

8.1.4.c Hybrid 3D-TOA/TDOA PC-Intel_Core_i7-2.7GHz_RAM16GB

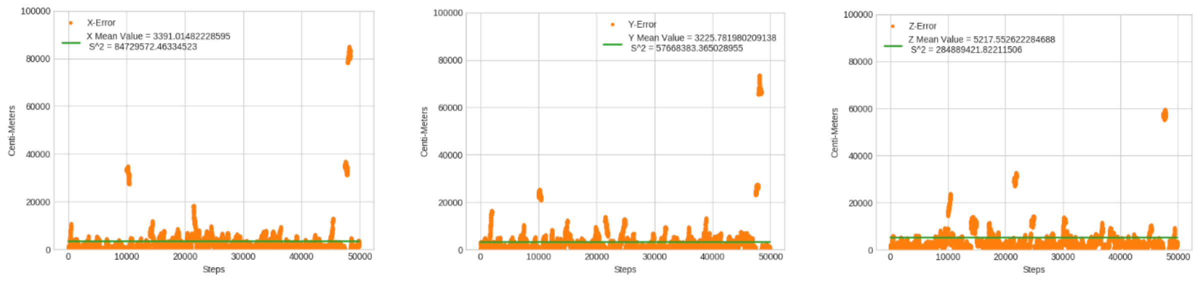


Figure 83. X, Y, Z Hybrid 3D-TOA/TDOA, PC-Intel_Core_i7-2.7GHz_RAM16GB

8.1.4.d Hybrid 3D-TOA/TDOA Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB

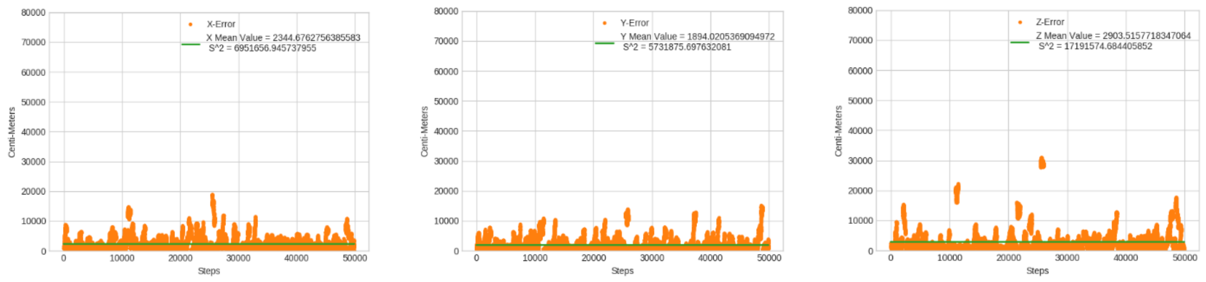


Figure 84. X, Y, Z Hybrid 3D-TOA/TDOA, Raspberry-Pi3ModelB1.4GHzRAM1GB

8.1.5 Chan method

From a sample of 50000 steps overall, we computed the mean error value for each coordinate x, y, z alongside with the standard deviation for Chan. We execute the simulation for four different environments as presented below.

8.1.5.a Chan PC-Intel_Core_i3-1.7GHz_RAM4GB

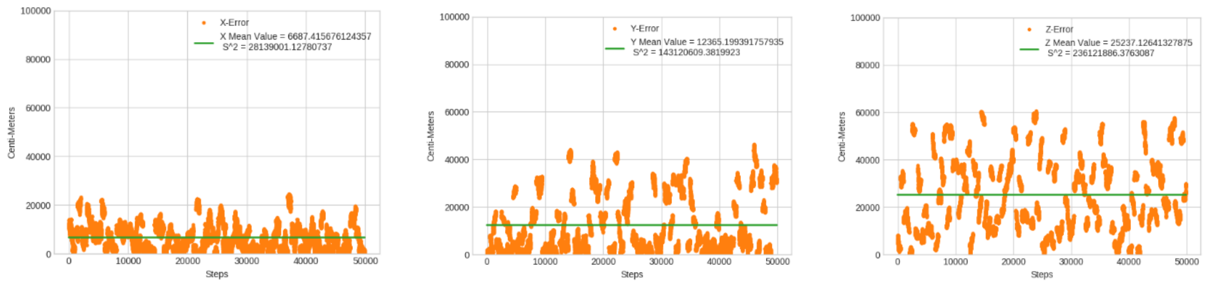


Figure 85. X, Y, Z Chan Alg, PC-Intel_Core_i3-1.7GHz_RAM4GB

8.1.5.b Chan PC-Inel_Core_i7-2.7GHz_RAM8GB

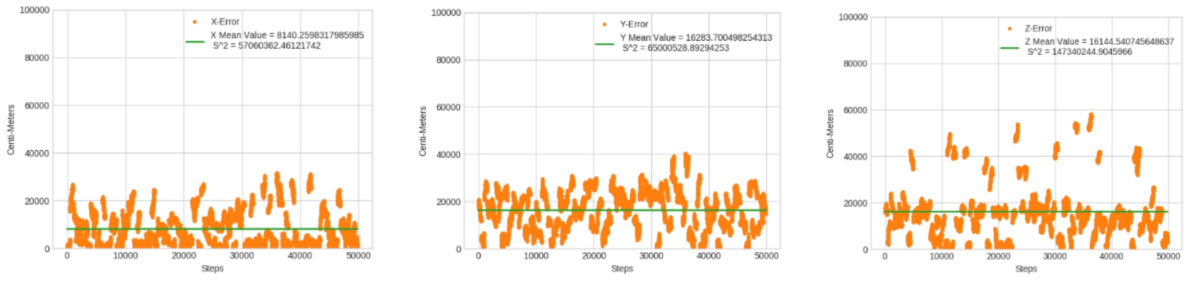


Figure 86. X, Y, Z Chan Alg, PC-Intel_Core_i7-2.7GHz_RAM8GB

8.1.5.c Chan PC-Intel_Core_i7-2.7GHz_RAM16GB

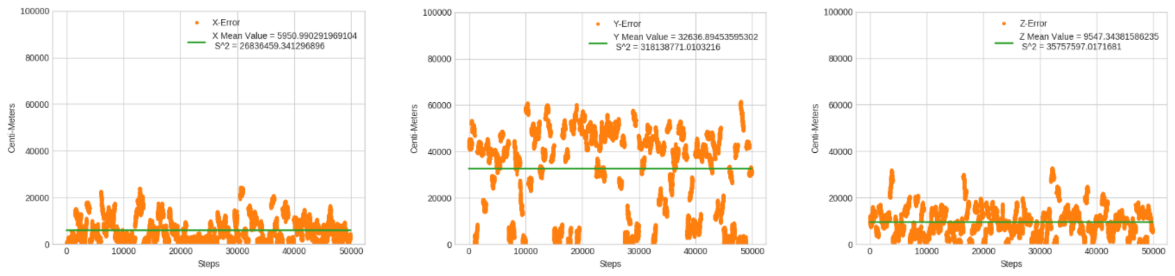


Figure 87. X, Y, Z Chan Alg, PC-Intel_Core_i7-2.7GHz_RAM16GB

8.1.5.d Chan Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB

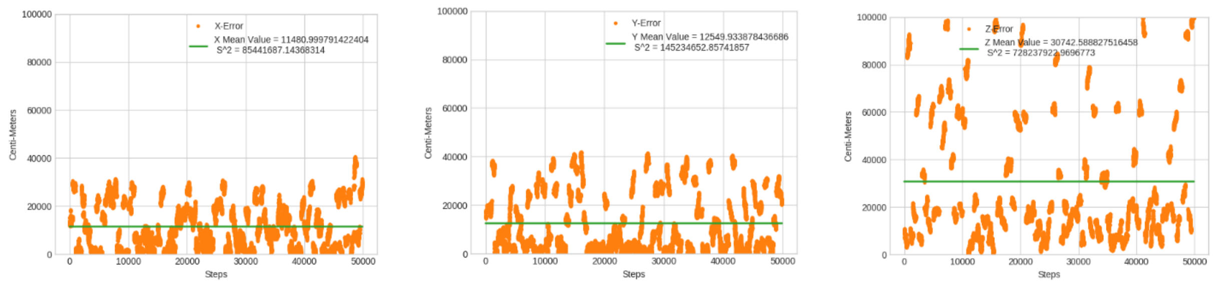


Figure 88. X, Y, Z Chan Alg, Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB

8.1.7 Performance Summary

Table 12. Performance Overview

Algorithm	Complexity	Accuracy	Minimum Nodes	Mean Error
3D-RSSI Multilateration	O(N)	10^{-9} meters	4	~10^{-10} meters
3D-TOA Multilateration	O(N)	10^{-9} meters	4	~10^{-10} meters
3D-TOA Direct	O(N)	10^{-9} meters	4	~10^{-10} meters
3D-RSSI Direct	O(N)	10^{-9} meters	4	~10^{-10} meters
Hybrid 3D-TOA/TDOA	O(N)	1-10 meters	4	~10 meters
Chan Method	O(N)	1-100 meters	5	~100 meters

8.2 Localization algorithms time validation

We performed time measurements for the same four environments as described above.

8.2.1 3D-RSSI Multilateration

For 50000 steps, we computed the mean execution time for all environments.

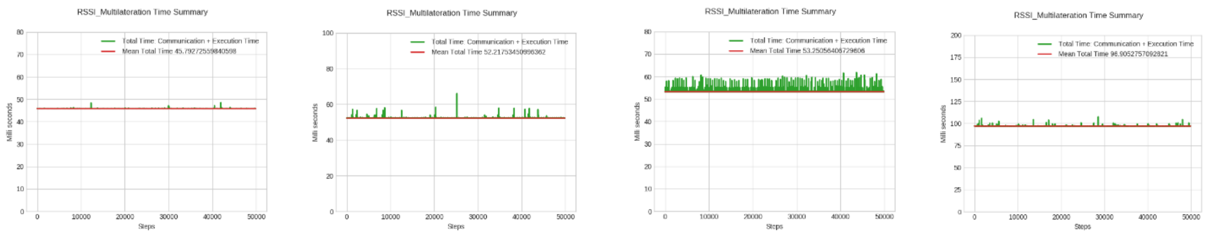


Figure 89. RSSI Multilateration method average execution time

8.2.2 3D-TOA Multilateration

For 50000 steps, we computed the mean execution time for all environments.

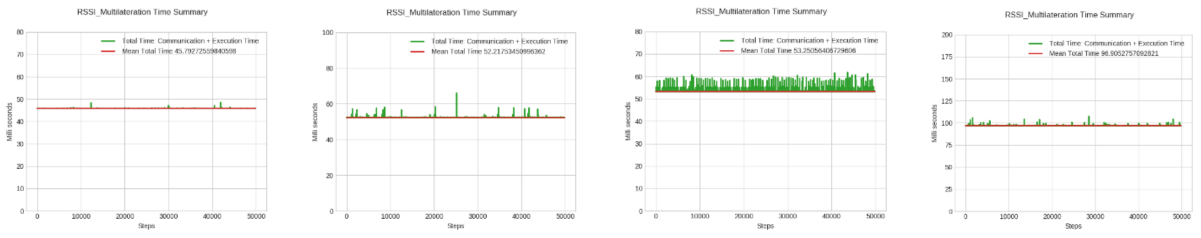


Figure 90. TOA Multilateration method average execution time

8.2.3 3D-RSSI Direct method

For 50000 steps, we computed the mean execution time for all environments.

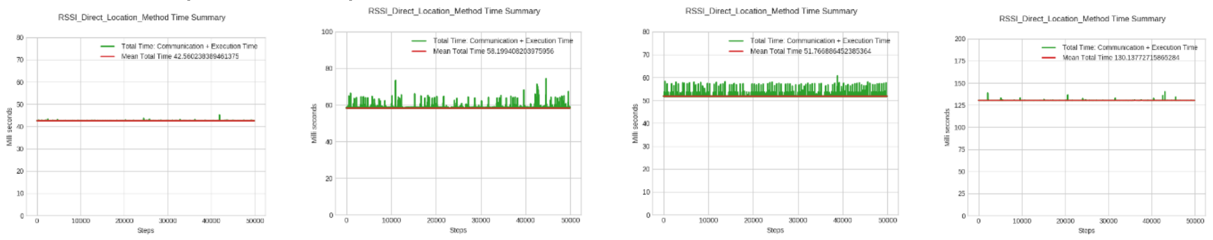


Figure 91. RSSI Direct method average execution time

8.2.4 3D-TOA Direct method

For 50000 steps, we computed the mean execution time for all environments.

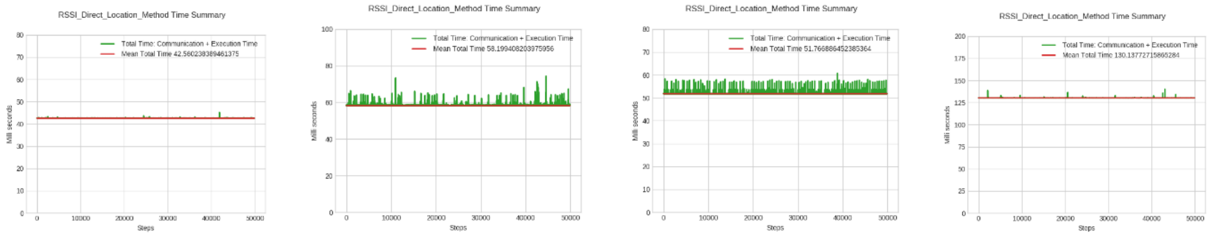


Figure 92. TOA Direct method average execution time

8.2.5 Hybrid 3D-TOA/TDOA

For 50000 steps, we computed the mean execution time for all environments.

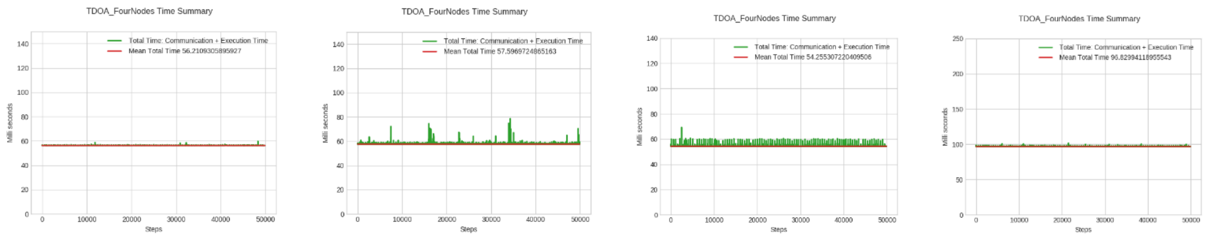


Figure 93. X, Y, Z error statistics TOA/TDOA Alg

8.2.6 Chan method

For 50000 steps, we computed the mean execution time for all environments.

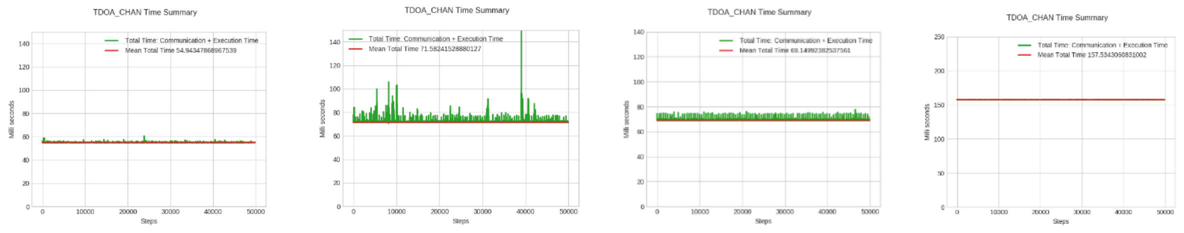


Figure 94. Chan method average execution time

8.2.7 Performance Summary

The overall execution, communication and overall time between all the nodes for all the executed algorithms is calculated:

Table 13. Execution Time Overview in milli seconds

Algorithm	PC-Intel_Core_i3-1.7GHz_RAM4GB			PC-Intel_Core_i7-2.7GHz_RAM8GB			PC-Intel_Core_i7-2.7GHz_RAM16GB			Raspberry-Pi_3_Model B+-1.4GHz_RAM1GB		
	Execution	Communication	Total	Execution	Communication	Total	Execution	Communication	Total	Execution	Communication	Total
3D-RSSI Multilateration	0.0427255	45.75	**45.79272**	0.055534	52.162	**52.217834**	0.069564	53.181	**53.25058408**	1.644275	96.795	**96.905275**
3D-TOA Multilateration	0.0427255	45.75	**45.79272**	0.055534	52.162	**52.217834**	0.069564	53.181	**53.25058408**	1.644275	96.795	**96.905275**
3D-RSSI Direct	0.043238	42.5169	**42.560238**	0.08442	58.115	**58.1994082**	0.0778864	51.689	**51.7668864**	0.166727158	129.971	**130.13772715**
3D-TOA Direct	0.043238	42.5169	**42.560238**	0.08442	58.115	**58.1994082**	0.0778864	51.689	**51.7668864**	0.166727158	129.971	**130.13772715**
Hybrid 3D-TOA/TDOA	0.1809305	56.03	**56.2109305**	0.201972	57.395	**57.596972**	0.18230722	54.073	**54.25530722**	0.664941185	96.165	**96.8299411**
Chan Method	0.958478	53.985	**54.943478**	1.15464	70.036	**71.582415**	1.353928	67.7959	**69.1499238**	5.43830608	152.096	**157.5346060**

The execution time is the calculated time of the algorithm's execution. The communication time is the calculated time of all the processes communication for the data exchange. The total is the sum of the execution and communication time.

8.3 Summary

Total time summary in milli seconds for all algorithms and environments.

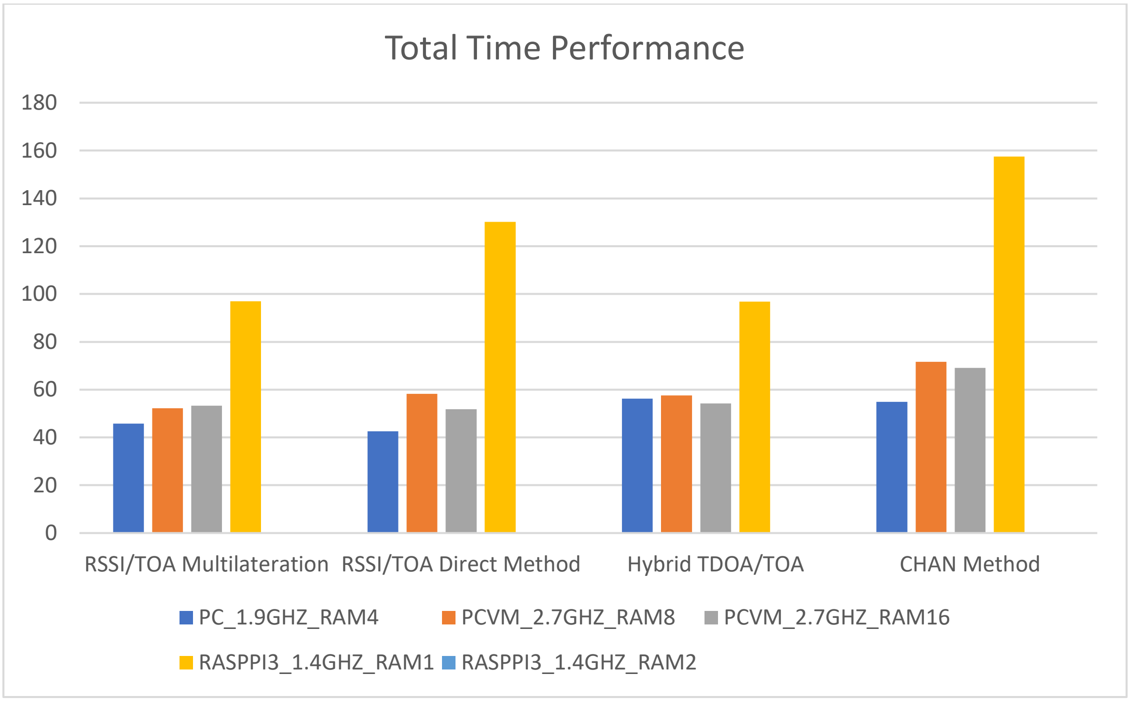


Figure 95. Total Time Performance summary

Total performance summary in meters for all algorithms and environments (power of 10)

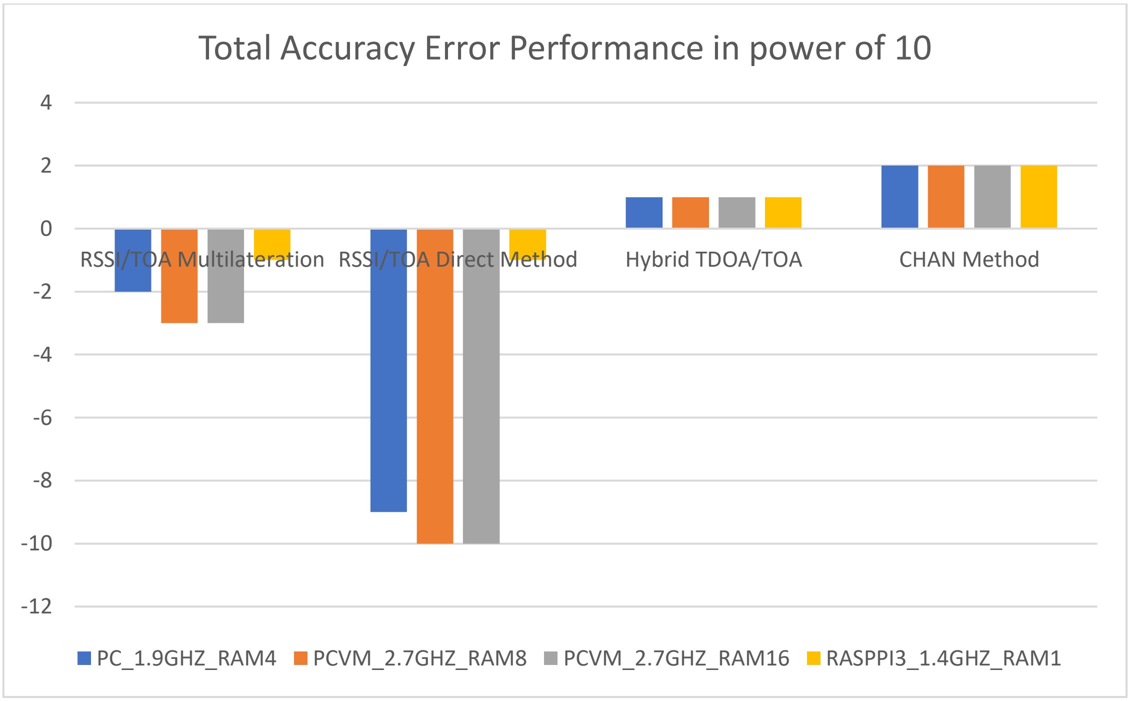


Figure 96. Total Accuracy Error Performance summary

9. CONCLUSION

In this paper, a prototype of Geo-location solution is developed and integrated into the Raspberry Pi3 platform. We proposed a Geo-location solution for low-power WSNs, using range-based GPS free state-of-the-art localization techniques in combination of a secure Hyper-visor. This approach allows to create a safe environment for each procedure of a node. In our case, we developed the procedure of localization. Localization is the most important in mobile WSNs. Without securing the topology of a network, no process can take place.

A compliance verification algorithm is developed and tested. Some preliminary results are reported and show the feasibility and the effectiveness of using trusted hyper-visor and range-based localization techniques. This solution might have a great impact on security and efficiency of a swarm. Future work includes more experiments, adding scalability, and the development of further services to be integrated into the platform.

Table 14. Algorithms Evaluation Summary

Summary	3D-RSSI/TOA Multilateration				3D-RSSI/TOA Direct Method				Hybrid 3D-TOA/TDOA				Chan Method			
	PCi 3.1.9GHz_16GB	PCi 7.2.7GHz_16GB	PCi 7.2.7GHz_16GB	RPi 3+1.4GHz_1GB	PCi 3.1.9GHz_16GB	PCi 7.2.7GHz_16GB	PCi 7.2.7GHz_16GB	RPi 3+1.4GHz_1GB	PCi 3.1.9GHz_16GB	PCi 7.2.7GHz_16GB	PCi 7.2.7GHz_16GB	RPi 3+1.4GHz_1GB	PCi 3.1.9GHz_16GB	PCi 7.2.7GHz_16GB	PCi 7.2.7GHz_16GB	RPi 3+1.4GHz_1GB
Calculated Error Performance	🟩	🟧	🟧	🟩	🟩	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Speed Performance	🟩	🟩	🟩	🟧	🟩	🟩	🟩	🟧	🟩	🟩	🟩	🟧	🟩	🟩	🟧	🟧
Overall	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟥	🟧	🟧	🟥	🟥

10. FUTURE WORK

Our future research goal is the implementation of the hypervisor XVisor in our proposed secure geo-localization system. Alongside the implementation will follow a series of security, speed, performance and pressure tests in the hypervisor in order to evaluate its reaction in several test environments. We will create a simulator for those purposes to simulate the environments.

After the evaluation then we can conclude if this hypervisor will satisfy the safe geo-localization system's requirements and if it can be utilized as the main factor of security and data protection in a geo-localization system. Our goal is to develop a secure geo-localization system with the usage of the most optimized and with high performance localization algorithm alongside with the utilization of the most secure and with high performance hypervisor for the creation of safe environments.

APPENDIX A

Hypervisors Implementation

[50] XMHF

Implementation

Requirements:

1. A TPM (v1.2): The BIOS feature is often called something like "embedded security device"
2. Virtualization extensions
    a. AMD: Secure Virtual Machine (SVM) or AMD Virtualization (AMD-V)
    b. Intel: Virtualization Technology (VT-x)
3. 2nd-level page tables Typically turned on implicitly along with Virtualization extensions, if the processor supports it.
    a. AMD: Nested Page Tables (NPT)
    b. Intel: Extended Page Tables (EPT)
4. Dynamic root of trust.
5. AMD: Late-launch (default with AMD-V)
6. Intel: Trusted Execution Technology (TXT). This feature is implemented partially by a signed software module, called an SINIT module. Some processors exist that have the TXT hardware support but do not (yet) have an SINIT module. Look for the

Building

Development Environment

XMHF and its Apps (e.g., TrustVisor) get built in a Linux environment with a recent version of gcc. XMHF has been verified to build on Ubuntu 10, 11, and 12 series, both 32 and 64-bit.

Build tools

A list of packages must be installed: **aptitude, install, pbuilder, texinfo, ruby, build-essential, autoconfli, btool, gcc-multilib** (for 64-bit platforms)

High-level Build Summary

One "drives" the build from `xmhf/xmhf`. The interesting high-level build commands include:

```
1. cd xmhf/xmhf
2. ./autogen.sh         # creates ./configure
3. ./configure          # creates Makefile from Makefile.in
4. make                 # Builds the selected hypapp and the XMHF core
5. make install         # Installs both binaries and dev headers and libs
6. make install-dev     # Installs just dev headers and libs
7. make test            # Runs various automated tests
8. make clean           # Deletes all object files
9. make init-late       # Explicitly builds the Linux kernel module for a dynamic late
   launch
```

The functioning of make install-dev and make test are hypapp-specific. For example, in TrustVisor, the primary prerequisite for tee-sdk and PAL development is having successfully run make install-dev in xmhf/xmhf.

How do I build an XMHF hypapp?

The preferred method for building different hypapps (e.g., TrustVisor, Lockdown) is by specifying which hypapp to build using `./configure`. The following describes the sequence of steps for building a XMHF hypapp using the helloworld hypapp as a running example.

Checkout the XMHF project source tree.

1. cd $WORK:
2. git clone git://git.code.sf.net/p/xmhf/xmhf xmhf
3. cd $WORK/xmhf/xmhf //Change working directory to the XMHF source tree root.
4. ./autogen.sh // Generate the `./configure` script.
5. ./configure --with-approot=src/example-hypapps/helloworld --with-apparchive=xmhfapp-helloworld.a //Configure the XMHF hypapp.
6. Generate/install binaries (default install path is specified with the `--prefix=` flag to `configure`).
    a. make
    b. make install
    c. make install-dev # optional (hypapp-specific)
    d. make test # optional (hypapp-specific)

Installation

Supported guest operating systems

In principle, any guest should be supported so long as it:

- Uses 'normal' 32-bit page tables. PAE is also supported on AMD. 64-bit is not yet supported.
- Does not use MTRRs. XMHF does not yet virtualize these (on Intel platforms), and an attempt by the guest to access MTRRs will trap and halt the system.

The following guest OSes are known to work:

- Windows XP
- Windows Server 2003
- Ubuntu 10.04 (with custom kernel to disable MTRRs)

Install XMHF binaries

If you have a .deb, use `dpkg -i` to install it. Otherwise, copy `init-x86.bin` and `hypervisor-x86.bin.gz` to `/boot`. You will need to install Grub 1, if you haven't already. On most modern Linux distributions, you will need to downgrade from Grub 2. On Windows machines without a Linux installation, you will need to install Grub.

Customize Linux Kernels

XMHF currently does not virtualize MTRRs (on Intel Platforms). On Linux, you will need to build or obtain a kernel with MTRR (`CONFIG_MTRR`) features disabled. Get a 2.6.32.X version, in this case we used 2.6.32.46. When making a new kernel yourself, do:

- make install //copies vmlinuz-2.6.32.46 into `/boot`
- make modules_install` //places modules in `/lib/modules/2.6.32.46`
- in `/boot`: mkinitramfs -o `initrd.img-2.6.32.46 2.6.32.46`

[53] UberXMHF

Implementation

Currently runs on:

1. Intel x86 32-bit hardware platform (pc-intel-x86_32)
2. Raspberry PI 3 ARMv8 32-bit hardware platform (rpi3-cortex_a53-armv8_32)

3. PC AMD x86 32-bit/Legacy Intel x86 32-bit hardware platforms (pc-legacy-x86_32)

Hardware Requirements

1. A TPM (v1.2 or above) The BIOS feature is often called something like "embedded security device"
2. Hardware Virtualization extensions Intel Virtualization Technology (VT-x)
3. Hardware Support for DMA Isolation Intel Virtualization Technology for Directed I/O
4. 2nd-level page tables: Intel Extended Page Tables (EPT)
5. Dynamic root of trust: Intel Trusted Execution Technology (TXT).

Supported OS

1. Ubuntu 16.04 LTS with Linux Kernel 4.4.x,
2. Ubuntu 12.04 LTS with Linux Kernel 3.2.0-27-generic and below

Installation

Make sure **your BIOS is up to date**, you could ruin your motherboard if your BIOS is buggy.

Configure target system to boot uberXMHF

You will need to install Grub 1, if you haven't already. On most modern Linux distributions, you will need to downgrade from Grub 2. The following commands accomplish the above task on Ubuntu:

```
1. sudo apt-get purge grub os-prober
2. sudo apt-get purge grub-gfxpayload-lists
3. sudo apt-get install grub
4. sudo update-grub
5. grub-install /dev/sda
```

And remove lines (if any) from “/boot/grub/menu.lst”:

```
• title      Chainload into GRUB 2
• root        b5912383-7f9e-4911-b51d-b14ce8cea70b
• kernel       /boot/grub/core.img
```

Get the correct SINIT module (Intel only)

uberXMHF launches itself with a dynamic root of trust. On Intel platforms, this requires a signed SINIT module provided by Intel, that matches your platform CPU and chipset.

Adding a Grub entry to boot Linux

You will need to add a Grub entry to `/boot/grub/menu.lst`. To ensure that it doesn't get clobbered, put it outside the AUTOMAGIC KERNEL LIST. When booting the machine, first choose the uberXMHF entry, and then choose a normal Linux entry. A grub entry for uberXMHF looks something like this:

```
1. title uberXMHF
2. rootnoverify (hd0,1)                              # should point to /boot
```

3. **kernel** /boot/xmhf-x86-vmx-x86pc.bin.gz serial=115200,8n1,0x3f8 # correct serial address
4. **modulenounzip** (hd0)+1 # should point to where grub is installed
5. **modulenounzip** /boot/4th_gen_i5_i7_SINIT_75.BIN # Intel TXT AC SINIT module

On Intel it is necessary to append one more line to provide the SINIT Authenticated Code module, or “ACmod”. This should be the last line. E.g.

- **module** /i5_i7_DUAL_SINIT_18.BIN

This will boot uberXMHF with debug output going to the specified serial port and then reload grub.

savedefault for unattended boot

Booting linux involves loading the grub menu twice. The first time you must select the uberXMHF entry, and the second time you must select an OS entry. You can automate this by using savedefault:

- **default** saved

Have your uberXMHF entry what you want as your default OS entry save each-other as the new default:

- **title** uberXMHF: savedefault 1
- **title** Default OS: savedefault 0

Verifying and Building

pc-intel-x86_32

For verification: Execute the following within the `uxmhf/` folder in the root tree of the sources:

1. Prepare for verification
   a. ./bsconfigure.sh
   b. ./configure --disable-debug-serial
   c. **make** uxmhf-verifyuobjs-prep
2. Verifying individual uberobjects
   a. **cd** xmhf-uobjs/<uobj-name>
   b. **make** verify
   c. **cd** ../..
   d. replace <uobj-name> with the uberobject directory name (e.g., `xh_hyperdep`)
3. Performing uberobject composition check: **make** uxmhf-verifyuobjs-compcheck
4. Verifying all the uberobjects: **make** uxmhf-verifyuobjs-all

For Building: Execute the following within the `uxmhf/` folder in the roottree of the sources:

1. Configure the serial debug output
   a. ./configure --enable-debug-serial=<your-serial-port-number>
   b. replace `<your-serial-port-number>` with the system serial port number. Note: if you omit the parameter to `--enable-debug-serial` the default port chosen is `0x3f8` or `COM1`.
2. Building the uberobject binaries and the final hypervisor image: **make** uxmhf-image

If everything goes well a final hypervisor image `xmhf-x86-vmx-x86pc.bin.gz` will be generated.

pc-legacy-x86_32

For verification: Use a combination of automated and manual techniques as described below:

1. OS: Ubuntu 10.10, 32-bit;
2. Verification Tools: CBMC: v4.1 32-bit;

Change working directory to the uberXMHF (pc-legacy-x86_32) source tree root:

1. **sudo** dpkg -i cbmc_4.1_i386.deb
2. **cd** ./xmhf
3. ./autogen.sh
4. ./configure --with-approot=hypapps/verify
5. **make** verifyall *or* **make** verify

make verifyall will perform full verification of the uberXMHF (pc-legacy-x86_32) core as well as the uberapp. Subsequently, you can use **make verify** to verify the uberapp assuming there are no further changes to the uberXMHF (pc-legacy-x86_32) core

For Building: A list of packages to install:

- **aptitude, install, pbuilder, texinfo, ruby, build-essential, autoconfli, btool, gcc-multilib** (for 64-bit platforms)

Change working directory to the uberXMHF (pc-legacy-x86_32) root directory.

1. **cd** ./xmhf
2. ./autogen.sh
3. ./configure --with-approot=hypapps/helloworld
4. **make**
5. **make** install
6. **make** install-dev # optional (hypapp-specific)
7. **make** test # optional (hypapp-specific)

Debugging

XMHF debugging is done primarily via the serial port. Use dmesg | grep ttyS on a Linux guest OS on the target system to examine the serial ports that the target system recognizes. For machines without a physical serial port (e.g., laptops), you may leverage Intel Active Management Technology (AMT) Serial-Over-LAN (SOL) capability.

AMT SOL exposes a serial port to the underlying platform once enabled (typically in the BIOS).

[58][59][60] TrustVisor

Implementation

The Trusted Execution Environment Software Development Kit. This is a set of tools and APIs for developing PALs and applications that use them. The current implementation is available in Github[ ]. There are all the executable and source files along with the instruction for the implementation and execution of the uberXMHF hypervisor, and also for XMHF and uxmhf-rpi.

TrustVisor Installation

To run TrustVisor on a given machine, installation is the same as for any other XMHFapp. See Installing uberXMHF for pc-legacy-x86_32 in the previous sessions. TrustVisor uses TPM NVRAM to securely store a master secret that is used to derive its cryptographic keys. For a real deployment of TrustVisor, this would need to be access controlled to TrustVisor's measurement, so that untrusted software would be unable to access this storage. [56]

TrustVisor Building

The TrustVisor build is primarily driven from the XMHF build process. When running `configure`, you will need to set `--with-approot=` to point to the TrustVisor source code. See Building uberXMHF for pc-legacy-x86_32 in the previous sessions.

TrustVisor Configurations

Disable the infineon driver

Modern Ubuntu has a tendency to load the Infineon-specific v1.1b TPM driver, when it should be using tpm_tis. Thus, we blacklist tpm_infineon. Don't forget to reboot after making this change. It is possible to manually remove this driver (`modprobe -r tpm_infineon`) and `modprobetpm_tis`, if you know what you'redoing. In "/etc/modprobe.d/blacklist.conf" add:

- **blacklist tpm_infineon**

Shut down trousers, if it is running

See if trousers is running first, shut down if necessary. It will probably start up again after the next reboot. You may wish to uninstall it or disable it more permanently if you're not otherwise using it.

1. ***/etc/init.d/trousers* status**
2. ***/etc/init.d/trousers* stop**

Install jTpmTools [61]

Install packet with the command below:

- **sudo** dpkg -i jtpmtools_0.6.deb

Set the tpm device to be accessible by jtss

1. **chown** jtss:tss /dev/tpm0
2. /etc/init.d/jtss **start**

3. /etc/init.d/jtss **status**
4. **cat** /var/log/jtss/tcs_daemon.log

Take ownership of the TPM

You will need to take ownership of the TPM, and set an owner password. It is important not to lose the owner password that you set. In TrustVisor's security model it is not security critical that the owner password is not compromised, so feel free to use a well known password or empty string if you are not using the TPM for other purposes that might require a strong TPM owner password.

- **jtttake_owner** -e ASCII -o 'owner_password'

Define the NV spaces

Define two nv spaces. One stores TrustVisor's master secret. The other stores the root of a hash chain used for replay protection (see [Memoir])

- **jtt** nv_definespace \
```
    --index 0x00015213 \
    --size 20 \
    -o 'owner_password' \
    -e ASCII \
    -p 11,12 \
    -w \
    --permission 0x00000000 \
    --writelocality 2 \
    --readlocality 2
```

- **jtt** nv_definespace \
```
    --index 0x00014e56 \
    --size 32 \
    -o 'owner_password' \
    -e ASCII \
    -p 11,12 \
    -w \
    --permission 0x00000000 \
    --writelocality 2 \
    --readlocality 2
```

Unload Linux TPM driver

Before running Trustvisor or PAL code that requires access to the NV RAM, we need to ensure the Linux TPM device driver is indeed removed. Hence, we want to stop all the drivers that rely on the Linux TPM.

1. **/etc/init.d/jtss stop**
2. **modprobe** -r tpm_tis

Installing TEE-SDK

On a machine where you are planning to develop PALs, you will also need to install the TrustVisor development headers. The tee-sdk currently expects those headers to be installed in two places. First, install the headers in a ‘normal’ system location. This can be installed by make install-dev, when you build TrustVisor. If you directly install TrustVisor binary on your platform without building it, please download and uncompress the uberXMHF package, go to the xmhf directory and run the following commands:

1. ./autogen.sh
2. ./configure --with-approot=hypapps/trustvisor
3. **make** install-dev

Second, you will then need to reconfigure to point to the Trustvisor PAL cross-compilation environment and install the headers again:

1. ./configure --with-approot=hypapps/trustvisor --prefix=$(SYSROOT)/usr
2. **make** install-dev

Note: $(SYSROOT) depends on your configuration of building TEE-SDK, see below for more details. The default $(SYSROOT) is /usr/local/i586-tsvc

Downloading and Patching Third Party Libraries

Before installing TEE-SDK, you need to download a few third-party libraries (e.g., newlib, openssl), and apply patches to them so that they could be used for PAL development. Download the:

1. **newlib-1.19.0.tar.gz**
   a. **cd** ../ports/newlib/newlib-1.19.0
   b. **patch** -p1 < ../newlib-tee-sdk-131021.patch
2. **openssl-1.0.0d.tar.gz**
   a. **cd** ../ports/openssl/openssl-1.0.0d
   b. **patch** -p1 < ../openssl-tee-sdk-131021.patch

Building and Installing TEE-SDK

After installing TrustVisor headers, downloading and patching third party libraries, go to TEE-SDK directory and run make to build and install TEE-SDK. If you would like to override the default paths, specify your overrides as parameters to make:

- **make PREFIX=$(PREFIX) HOST=$(HOST) SYSROOT=$(SYSROOT)**
- **$(PREFIX)** specifies where you will install various utilities, libraries, and headers. The default **$(PREFIX)** is /usr/local.
- **$(HOST)** is the host-name to use for PAL code. The default $(HOST) is i586-tsvc.
- **$(SYSROOT**) points to the path where libraries to be linked against PAL code will be installed. The default $(SYSROOT) is $(PREFIX)/$(HOST)

Of course, you may install each tee-sdk component individually, either by specifying a target to make, or by manually performing the steps in the corresponding make recipe. At the time of this writing, the components installed by make are:

1. **toolchain**: these are wrappers to utilities such as gcc, with names like i586-tsvc-gcc. They mostly serve to override the system paths with paths in $(SYSROOT).
2. **tz**: This implements the TrustZone API for managing and communicating with services (pals) running the trusted execution environment (trustvisor).
3. **newlib**: this is an implementation of libc targeted for PALs. Functions that do not involve IO should work as expected. IO functions currently fail gracefully. The toolchain i586-tsvc-gcc will link against this library by default, unless -nostdlib is used.
4. **openssl**: This is the well-known openssl library, ported for use with pals. It is not installed by default, but can be installed with make openssl

Using TEE-SDK

Compiling applications

The TEE-SDK installs several libraries to the development machine. There is a front-end library for applications (tee-sdk-app), a front-end library for services (tee-sdk-svc), and for each device there are application and service back-end libraries (tee-sdk-app-devname and tee-sdk-svc-devname). We use **pkgconfig** to simplify management of these libraries. The compile time flags needed to link against a package can be obtained using **pkg-config --cflags packagename**.

The linking flags can be obtained using pkg-config --libs --static packagename. Note that we only support static linking for now. If you installed tz to a non-standard location $tzinstallprefix, you may need to set PKG_CONFIG_LIBDIR to include $tzinstallprefix/lib/pkgconfig. An application using the tee-sdk to communicate with a service running in a trusted environment must link against at least one application back-end. It is also permissable to link against multiple back-ends; a single application can communicate with services running on multiple devices.

Compiling services (PALs)

You must compile and link using exactly one service back-end package. At the time of this writing, there is only one anyways: tee-sdk-svc-tv. pkgconfig will automatically pull in the service front-end tee-sdk-svc as a dependency. Using the compile and link flags from those packages is important not only to link against the corresponding libraries; they also reference compiler options to eliminate code-constructs that are unsupported inside services, and linker options to ensure the necessary layout in the final binary.

Services to be run under TrustVisor need to be compiled somewhat specially. A PAL is linked together into the same binary with the application that runs it. At run-time, the application registers the PAL with TrustVisor. Using the raw TrustVisor interfaces for PAL management, you would need to keep track of which address ranges belong to

PAL code, data, etc., and make sure those sections are page-aligned. Things can get tricky if you want some code to be accessible to both the PAL code and the application code, and trickier still if you want to use different implementations for the same function in PAL and application code (such as linking the PAL against a version of libc that does not make system calls while linking the regular code with the standard version of libc).

The TEE-SDK has some tools to take care of these details for you. The basic approach is use partial linking to link all PAL code into a single object file (.o), rewrite all symbols except for the PAL entry-point in that object file to be private, and then use a linker script to link this object file with the regular application while mapping the code and data of the PAL to special page-aligned sections. The TrustVisor back-end provides simplified functions for registering a PAL that has been built and linked this way.

The TEE-SDK includes pkg-config files that specify the necessary compilation and link flags, and Makefile snippets that can be included in your own Makefiles to automate most of the process. Pointing your makefile at those makefile snippets and\or pkg-config files (rather than copying and modifying a monolithic Makefile with these things hard-coded) will help keep your pal up to date as the build process evolves. See examples/newlib/Makefile for a good starting point of a Makefile that dynamically incorporates the TEE-SDK-provided Makefile snippets and pkg-config files.

Compiling and running the test example

After installation in tz, you should be able to compile and run the test example in ../examples/test. Remember to set the PKG_CONFIG_LIBDIR environment variable if you installed to a non-system directory.

Loading and unloading services

Services are loaded and unloaded through the TrustZone service manager. The TrustVisor back-end provides some convenience functions for an application to load an unload a single PAL:

APPENDIX B

Sequencing in Three-Dimensional Grids python

The sequencing and computation of the transition matrix for the bordered symmetric random walk.

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import random
import math
import numpy as np
from mpl_toolkits.mplot3d import Axes3D, art3d
from urllib3.connectionpool import xrange

def nodeDistancesCalculation (p1, p2):
        dist = math.sqrt((p2[0] - p1[0]) ** 2 + (p2[1] - p1[1]) ** 2 + (p2[2] - p1[2])**2)
        return dist

def pointInsideTetrahedron (v1,v2,v3,v4,p):
        def tetraCoord_Dorian(A, B, C, D):
                v1 = B - A;v2 = C - A; v3 = D - A
                mat = np.array((v1, v2, v3)).T
                M1 = np.linalg.inv(mat)
                return (M1)
        M1=tetraCoord_Dorian(v1,v2,v3,v4)
        newp = M1.dot(p-v1)
        return (np.all(newp>=0) and np.all(newp <=1) and np.sum(newp)<=1)

def brownian_motion_simulation (xyz,cur):
        m = 3; n = 500; d = 1000.0;t = 1.0
        dt = t / float(n - 1)
        for j in range(1, n):
                s = np.sqrt(2.0 * m * d * dt) * np.random.randn(1)
                dx = np.random.randn(m)
                norm_dx = np.sqrt (np.sum(dx ** 2))
        for i in range (0, m):
                dx[i] = s * dx[i] / norm_dx
                cur[0] += dx[0]
                cur[1] += dx[1]
                if cur[2] + dx[2] > 0:
                        cur[2] +=  dx[2]
                else:
                        cur[2] += abs(dx[2])
                p = np.array(cur[:])
                xyz.append(cur[:])

mpl.rcParams['legend.fontsize'] = 10
fig = plt.figure()
ax = fig.gca(projection='3d')
xyz = []
cur = [0,-20, 20]
```

```
A = np.array ([10, -40, 0])
B = np.array ([-30, 30, 0])
C = np.array ([30, 30, 0])
D = np.array ([0, 0, 30])
brownian_motion_simulation(xyz,cur)
x, y, z = zip (*xyz)

file = open("TraceOutput.txt","w")
header = " A = " + str(A) + "\n B = " + str (B) + "\n C = " + str (C) + "\n D = " + str (C) + "\n\n"
file.write(header)
file.write("Node E Trace Log: \n\n")
for i in range(0 , len(x)):
            inputFile = "[ " + str (x[i]). ljust (10)[:15] + ", " + str (y[i]). ljust (10)[:15]   + ", " + str (z[i]). ljust
(10) [:15]  + "] :\t\t Distances-> AE = " + str (nodeDistancesCalculation(A,[x[i],y[i],z[i]])). ljust (10)[:15] +
", BE = " + str (nodeDistancesCalculation (B,[x[i],y[i],z[i]])). ljust (10)[:15] + ", CE =  "+ str
(nodeDistancesCalculation (C,[x[i],y[i],z[i]])). ljust (10)[:15]+ ", DE =  " + str
(nodeDistancesCalculation(D,[x[i],y[i],z[i]])).ljust (10)[:15] + "\n"
            file.write(inputFile)
ax.plot(x, y, z,'C3', label='E node movement')
ax.scatter(cur[0], cur[1], cur[2],'C7',label="End")
ax.scatter(x[0], y[0], z[0],'C3',label="Start")
v = np.array([A, B, C, C ,D ])
ax.scatter3D(v[:, 0], v[:, 1], v[:, 2])
verts = [ [v[0],v[1],v[4]], [v[0],v[3],v[4]],[v[2],v[1],v[4]], [v[2],v[3],v[4]], [v[0],v[1],v[2],v[3]]]
ax.add_collection3d(Poly3DCollection(verts, facecolors='cyan', linewidths=1, edgecolors='r', alpha=.25))
ax.scatter(x[-1], y[-1], z[-1], c='b', marker='o')
ax.legend()
plt.show()
```

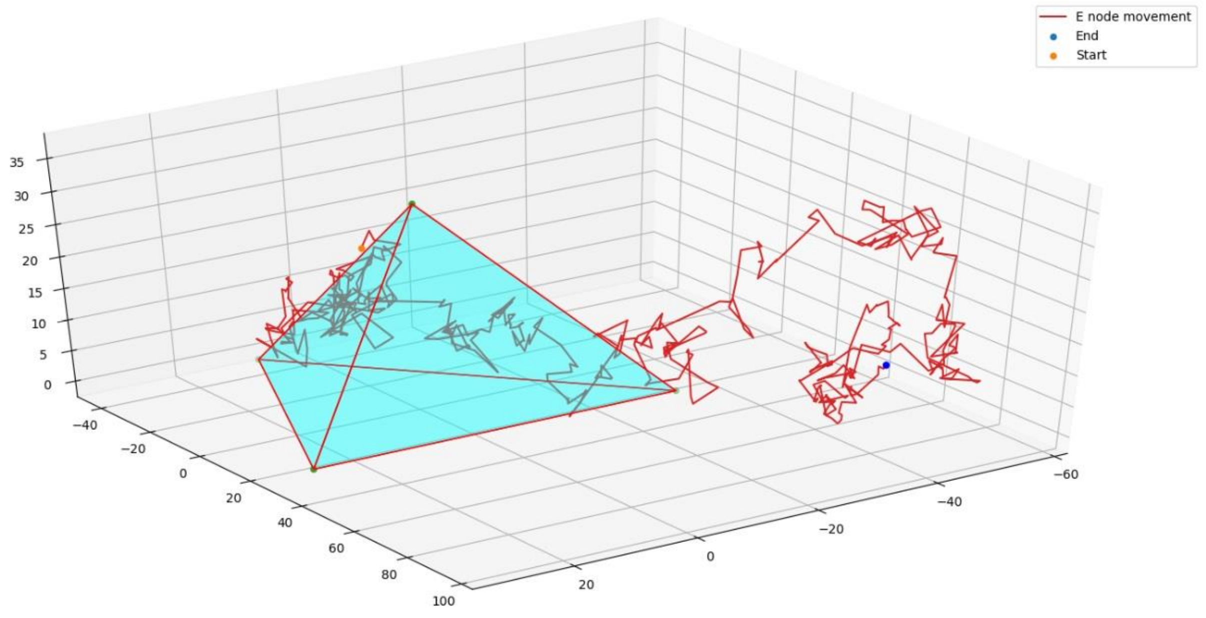


Figure 97. Random Walk output example

APPENDIX C

Intersection points between 3 spheres python

```
function intersect3spheres (x1, x2, x3, y1, y2, y3, z1, z2, z3, r1, r2, r3) {
        var a1, b1, c1, k1, a3, b3, c3, k3, a31, b31, e, f, g, h, A, B, C, x, y, z, x_, y_, z_, rootD;
        k1 = r1 * r1 - r2 * r2 - x1 * x1 + x2 * x2 - y1 * y1 + y2 * y2 - z1 * z1 + z2 * z2;
        a1 = 2 * (x2 - x1);
        b1 = 2 * (y2 - y1);
        c1 = 2 * (z2 - z1);
        k3 = r3 * r3 - r2 * r2 - x3 * x3 + x2 * x2 - y3 * y3 + y2 * y2 - z3 * z3 + z2 * z2;
        a3 = 2 * (x2 - x3);
        b3 = 2 * (y2 - y3);
        c3 = 2 * (z2 - z3);
        if (a1 === 0) {
                e = -c1 / b1;
                f = k1 / b1;
        } else if (a3 === 0) {
                e = -c3 / b3;
                f = k3 / b3;
        } else {
                a31 = a3 / a1;
                e = - ((a31 * c1 - c3) / (a31 * b1 - b3));
                f = (a31 * k1 - k3) / (a31 * b1 - b3);
        }
        if (b1 === 0) {
                g = -c1 / a1;
                h = k1 / a1;
        } else if (b3 === 0) {
                g = -c3 / a3;
                h = k3 / a3;
        } else {
                b31 = b3 / b1;
                g = - ((b31 * c1 - c3) / (b31 * a1 - a3));
                h = (b31 * k1 - k3) / (b31 * a1 - a3);
        }
        A = g * g + e * e + 1;
        B = -x1 * g - y1 * e - 2 * z1 - x1 * g - y1 * e + 2 * g * h + 2 * e * f;
        C = x1 * x1 + y1 * y1 + z1 * z1 - 2 * x1 * h - 2 * y1 * f + h * h + f * f - r1 * r1;
        rootD = Math.sqrt(B * B - 4 * A * C);
        z = (-B + rootD) / (2 * A);
        z_ = (-B - rootD) / (2 * A);
        x = g * z + h;
        x_ = g * z_ + h;
        y = e * z + f;
        y_ = e * z_ + f;
        return [x, y, z, x_, y_, z_];
}
```

APPENDIX D

Random walk mobility with R

```
caculateTransitionMatrix3d = function (width , depth , height ) {
        w = width d = depth h = height
        numberOfStates = w∗d∗h
        P = matrix (0, nrow = numberOfStates, ncol = numberOfStates )
        nodePoints = matrix ( 0 , nrow = numberOfStates , n col = 3 )
        nodeNumber = 0
        for (z in 1: h) {for (y in 1: d) {for (x in 1: w) {
                                nodeNumber = nodeNumber + 1
                                nodePoints [ nodeNumber , ] = c (x , y , z )
        }}}
        calculateNeighbourNodes = function (x , y , z ) {
                neighbourNodes = matrix (nrow = 0 , n col = 3)
                if ( x > 1 ) neighbourNodes = rbind ( neighbourNodes , c ( x−1 , y , z ) )
                if ( x < w ) neighbourNodes = rbind ( neighbourNodes , c ( x+1 , y , z ) )
                if ( y > 1 ) neighbourNodes = rbind ( neighbourNodes , c ( x , y−1 , z ) )
                if ( y < d ) neighbourNodes = rbind ( neighbourNodes , c ( x , y+1 , z ) )
                if ( z > 1 ) neighbourNodes = rbind ( neighbourNodes , c ( x , y , z−1 ) )
                if ( z < h ) neighbourNodes = rbind ( neighbourNodes , c ( x , y , z+1 ) )
                return ( neighbourNodes )
        }
        neighbourNodes = list( )
        nodeDegrees = c ( )
        nodeNumber = 0
        for (z in 1: h) {for (y in 1: d) {for (x in 1: w) {
                                nodeNumber = nodeNumber + 1
                                neighbourNodes [ [ nodeNumber ] ] =calculateNeighbourNodes (x,y ,z)
                                nodeDegrees[[nodeNumber]]=nrow( neighbourNodes [[nodeNumber]] )
        }}}
        nodeTransformation = function (x, y, z) {return ( x + w ∗ (y−1) + w ∗ d ∗ ( z −1) )}
        transformedNeighbourNodes = list ()
        nodeNumber = 0
        for (z in 1: h) {for (y in 1: d) {for (x in 1: w) {res = c ( )
                                nodeNumber = nodeNumber + 1
                                for (i in 1: nrow ( neighbourNodes [ [ nodeNumber ] ] ) ) {
                                        value1 = neighbourNodes [ [ nodeNumber ] ] [ i , 1 ]
                                        value2 = neighbourNodes [ [ nodeNumber ] ] [ i , 2 ]
                                        value3 = neighbourNodes [ [ nodeNumber ] ] [ i , 3 ]
                                        res = c (res, nodeTransformation (value1,value2 , value3 ) )
                                }
                                TransformedNeighbourNodes [ [ nodeNumber ] ] = res
        }}}
        for ( nodeNumber in 1 : numberOfStates ) {
           P [ nodeNumber , transformedNeighbourNodes [ [ nodeNumber ] ] ] = 1 / nodeDegrees[
        nodeNumber ]
        }
        return ( P )
}
```

APPENDIX E

Way points output example

```
 [ -1.762666719825, -22.60864158980, 18.895749148212] :Distances-> AE = 28.246575218699, BE = 62.626396164657, CE =  64.292966201223, DE =  25.249991197641

[ -1.435704363900, -20.07952591316, 18.227263374445] : Distances-> AE = 29.322921959861, BE = 60.465783969079, CE =  61.874029727047, DE =  23.320504616597

[ -1.635140021379, -19.78191325741, 18.451133156449] : Distances-> AE = 29.742088521229, BE = 60.193425587293, CE =  61.801822679420, DE =  22.964627155337

[ -2.170852646725, -16.56715680376, 19.083690396634] : Distances-> AE = 32.579365149776, BE = 57.507814898071, CE =  59.729817444538, DE =  19.958684854737

[ -0.911234804309, -14.79329197055, 16.260585835480] : Distances-> AE = 31.919270454529, BE = 55.830116584891, CE =  56.800969132575, DE =  20.209981142468

[ -2.933428126028, -16.52180310888, 16.717485224969] : Distances-> AE = 31.590720229147, BE = 56.359132265832, CE =  59.400026640406, DE =  21.400938701098

[ -3.233030907304, -16.21198961408, 17.072244905087] : Distances-> AE = 32.131668042807, BE = 56.066747405224, CE =  59.425952861336, DE =  20.985898786779

[ -8.211052570226, -14.60776864022, 18.067570130518] : Distances-> AE = 36.094943407144, BE = 52.830373305313, CE =  61.452214378376, DE =  20.571610815576

[ -7.066307283342, -15.24110842872, 16.203182664678] : Distances-> AE = 34.158522869004, BE = 53.247115245307, CE =  60.689473188916, DE =  21.738819042053

[ -7.215896947749, -19.73667313454, 17.275849294304] : Distances-> AE = 31.708744727779, BE = 57.369913502002, CE =  64.476465543306, DE =  24.566429255419

[ -6.510211581422, -19.52333330373, 17.108235036611] : Distances-> AE = 31.377901005664, BE = 57.419704000717, CE =  63.861160319091, DE =  24.284583628194

[ -6.921649413950, -19.21862076507, 16.982969637673] : Distances-> AE = 31.727420309705, BE = 56.951770415875, CE =  63.828693259197, DE =  24.222049751724

[ -8.608554716747, -19.87745089706, 16.712622340045] : Distances-> AE = 32.101511436801, BE = 56.786140780799, CE =  65.250995017596, DE =  25.412096992472

[ -9.237954874666, -17.28234388679, 15.501404815222] : Distances-> AE = 33.563139875210, BE = 53.916380743341, CE =  63.368215198322, DE =  24.376802145069

[ -8.017354464469, -14.73585393614, 14.408447230251] : Distances-> AE = 34.212651020066, BE = 51.885804260165, CE =  60.450138291492, DE =  22.902399036091

[ -7.773636624478, -13.87057299672, 12.780614015330] : Distances-> AE = 34.088021488307, BE = 50.813211850567, CE =  59.285908051644, DE =  23.437778805082

[ -7.756594041074, -13.87168749046, 12.810773421726] : Distances-> AE = 34.089606366140, BE = 50.829223755406, CE =  59.282385853693, DE =  23.410638079947

[ -6.318883068392, -14.81175210539, 13.297957130046] : Distances-> AE = 32.826657466645, BE = 52.399657343502, CE =  59.194510369827, DE =  23.200743946983

[ -5.847657978464, -15.31094916685, 12.753802494379] : Distances-> AE = 31.989951117649, BE = 52.906306028603, CE =  59.167526355324, DE =  23.791838866006

[ -4.854403492534, -14.67398669539, 13.818364886816] : Distances-> AE = 32.450076443196, BE = 53.094381238712, CE =  58.322737746319, DE =  22.377140875034

[ -5.779379780283, -13.77367476000, 12.496636681137] : Distances-> AE = 33.060170752501, BE = 51.564900602668, CE =  57.900471049872, DE =  23.010499223427

[ -6.789397403698, -13.64930767381, 13.007996080657] : Distances-> AE = 33.844509352077, BE = 51.119488410208, CE =  58.548525034925, DE =  22.828221881584

[ -6.807716379025, -13.38795293836, 12.618780357379] : Distances-> AE = 33.910676731616, BE = 50.790059041561, CE =  58.279979949621, DE =  22.971484114922

[ -9.410265048017, -15.63417315707, 15.238292503697] : Distances-> AE = 34.679352147958, BE = 52.331830691756, CE =  62.192059857451, DE =  23.471013331034

[ -7.967557587745, -15.16523952138, 13.868788797373] : Distances-> AE = 33.644342115803, BE = 52.131283145339, CE =  60.611695182605, DE =  23.530457677631

[ -6.722832997596, -14.69323988697, 13.073492032766] : Distances-> AE = 33.030311000877, BE = 52.059853910503, CE =  59.304033158753, DE =  23.400736181891

[ -5.853318098042, -17.47807643688, 10.180947461185] : Distances-> AE = 29.363522045354, BE = 54.229896553730, CE =  60.359587904436, DE =  27.065463828035

[ -6.780783829204, -17.74870978155, 10.340784832722] : Distances-> AE = 29.726191368294, BE = 54.092542150894, CE =  61.153063503399, DE =  27.340089049629

[ -10.54429840251, -17.40045290905, 13.720079936016] : Distances-> AE = 33.482059659135, BE = 53.043075458394, CE =  63.866138621230, DE =  26.057509234982

[ -10.63939466325, -17.29566405599, 13.833632412808] : Distances-> AE = 33.657701473164, BE = 52.944142860522, CE =  63.873387438170, DE =  25.955503364464

[ -10.39633501875, -18.13911368389, 13.448207606599] : Distances-> AE = 32.783580044790, BE = 53.689218981076, CE =  64.265795233924, DE =  26.665953170544

[ -11.06415834911, -20.09595883895, 14.563396875925] : Distances-> AE = 32.433965993428, BE = 55.500123599748, CE =  66.392489947892, DE =  27.650531236151

[ -11.42314336957, -19.97983844946, 13.694994445045] : Distances-> AE = 32.362181836097, BE = 55.051219119579, CE =  66.343152856197, DE =  28.205236305616

[ -9.162960293850, -18.93527464371, 15.098801900094] : Distances-> AE = 32.232212464826, BE = 55.288490185478, CE =  64.469933940185, DE =  25.778870650054

[ -8.827541078686, -20.22569446024, 12.771617624540] : Distances-> AE = 30.143219459715, BE = 55.982208044487, CE =  64.755791609543, DE =  27.996809767725

[ -15.18670726571, -16.28999650908, 13.976003642759] : Distances-> AE = 37.307682402560, BE = 50.571989245680, CE =  66.181046895245, DE =  27.436627400197

[ -14.86539710722, -14.94547378950, 13.853011549237] : Distances-> AE = 37.919957620175, BE = 49.407061720649, CE =  64.999272309274, DE =  26.553200446975
```

ACRONYMS

AIK	Attestation Identity Key
ALB-DRM	Authenticated Location based on DRM
APIT	Approximate Point In Triangle
AOA	Angle Of Arrival
AWS	Authenticated Weight-based
CPE	Convex Position Estimation
CPU	Central Processing Unit
CSLT	Collaborative Secure Localization Trust
DB	Data Base
DCP	Degree of CoPlanarity
DFPLE	Distributed Fermat-Point Location Estimation
DLP	Discrete Logarithm Problem
DRTM	Dynamic Root of Trust Management
DV hop	Distance Vector
DoS	Denial of Service Attacks
dB	decibel
ECC	Elliptic Curve Cryptography
FP-MPP-APIT	Fermat Point Mid Perpendicular Plane APIT
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IDCP	Improved Degree of CoPlanarity
IEC	International Electrotechnical Commission
IFP	Integer Factorization Problem
IOMMU	IO Memory Management Unit
IP	Instruction Pointer
ISO	International Organization for Standardization
LDEA	Location-Dependent Encryption Algorithm

MAINV	Mutual Authentication Insider Node Validation
MMSE	Minimum Mean Square Error
MMU	Memory Management Unit
MPA	Multilateral Privacy Algorithm
MPRSA	Mean Power with Rivest-Shamir-Adelman
μTPM	μ(micro)-Trust Platform Module
NPT	Nested Page Table
OWR	One Way Ranging
PAL	Piece of Application Logic
PCR	Platform Configuration Registers
PPT	Protection Page Table
RAM	Random Access Memory
RF	Radio Frequency
RSA	Rivest–Shamir–Adleman
RSSI	Received Signal Strength Indication
S. DV-HOP	Secure DV-Hop
TCB	Trusted Computing Base
TD	Tolerance Distance
TDOA	Time Difference Of Arrivals
TGS	Trust Geolocation Server
TOA	Time Of Arrival
TOA-ECC	TOA - Elliptic Curve Cryptography
TPM	Trust Platform Module
TPMBG	TPM Based Geo-location
TRTM	TrustVisor Root of Trust for Measurement
TWR	Two Way Rangin

REFERENCES

[1] Geo-location Introduction. https://en.wikipedia.org/wiki/Geo-location

[2] Where On Earth Identifier: https://en.wikipedia.org/wiki/WOEID

[3] Natural Area Code: https://en.wikipedia.org/wiki/Natural_Area_Code

[4]"GPS-Free Localization Algorithm for Wireless Sensor Networks" Lei Wang, Qingzheng Xu

[5]"A Decentralized Architecture for Simultaneous Localization and Mapping" Ehsan Asadi and Mohammad Bozorg

[6] N.A. Carlson, "Federated filter for computer-efficient, near-optimal GPS integration." *IEEE Position Location and Navigation Symposium* (PLANS), pp. 306-314, 1996.

[7] S. Grime, and H. F. Durrant-Whyte, "Data fusion in decentralized sensor networks." *Control Engineering Practice*, vol. 2, pp. 849-863,1994.

[8] E. Nebot, M. Bozorg, and H. Durrant-Whyte, "Decentralized Architecture for Asynchronous Sensors," *Autonomous Robots*, pp.147-164, 1999.

[9] E. Nettelton, S. Thrun, H. Durrant-Whyte, and S. Sukkarieh, "Decentralized SLAM with low-bandwidth communication for teams of vehicles." *4th International Conferences on Field and Service Robotics*, *Japan*, 2003.

[10] "Range-free and range-based localization of wireless sensor networks" Xiao,Qingjun, 2011

[11] "Range Free Localization Techniques in Wireless Sensor Networks: A Review", Santar Pal Singh S.C.Sharma 2015

[12] Veris White Paper "VerisAerospond Wireless Sensors: Received Signal Strength Indicator (RSSI)"

[13] ECE Senior Capstone Project 2017 Tech Notes "Finding Location with Time of Arrival and Time Difference of Arrival Techniques"

[14] "Angle of Arrival Localization for Wireless Sensor Networks" Rong Peng and Mihail L. Sichitiu Department of Electrical and Computer Engineering

[15] "Analysis of DV-Hop Localization Algorithm in Wireless Sensor Networks" Mehak Khurana, Ashish Payal 2011

[16] "Improved APIT localization algorithm in wireless sensor networks", Shubhankar Jain Akanksha Singh ; Amanpreet Kaur ; Shikha Jain 2017

[17] "New Wireless Sensor Network Localization Algorithm for Outdoor Adventure", XUEJIAN ZHAO, XINHUI ZHANG, ZHIXIN SUN AND PAN WANG. 2018

[18] "An Improved 3D Localization Algorithm for the Wireless Sensor Network" Yan Xu 2015

[19] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices", IEEE Personal CommunicationsMagazine, 7(5):28–34, October 2000.

[20] "Novel Centroid Localization Algorithm for Three-Dimensional Wireless Sensor Networks" Hongyang Chen, Pei Huang, Marcelo Martins, H.C. So 2008

[21] "Estimation of Distributed Fermat-Point Location for Wireless Sensor Networking" Po-Hsian Huang 1,*, Jiann-Liang Chen 2 , Yanuarius Teofilus Larosa 2 and Tsui-Lien Chiang 2011

[22] L. Doherty, K. S. J. Pister, and L. E. Gaoui, “Convex Position Estimation in Wireless Sensor Networks”, Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies, vol.3, (2001) April, pp. 1655-1633.

[23] “An Improved CPE Localization Algorithm for Wireless Sensor Networks” Jianmin Zhang, Hua Li and Jian Li 2015

[24] “A Hybrid DV-Hop Algorithm Using RSSI for Localization in Large-Scale Wireless Sensor Networks” Omar Cheikhrouhou , Ghulam M. Bhatti and Roobaea Alroobaea

[25] “An Enhanced Hybrid 3D Localization Algorithm Based on APIT and DV-Hop”, Lianjun Yi, Miaochao Chen 2017

[26] “Time Difference of Arrival (TDoA) Localization Combining Weighted Least Squares and Firefly Algorithm” Peng Wu, Shaojing Su, Zhen Zuo *, Xiaojun Guo, Bei Sun,Xudong Wen 2019

[27] Bucher, R.; Misra, D. A Synthesizable VHDL Model of the Exact Solution for Three-dimensional Hyperbolic Positioning System. VLSI Desgin 2002, 15, 507–520. [CrossRef]

[28] 3D Tdoa Problem Solution with Four Receiving Nodes, Javier Díez-González, Rubén Álvarez, Lidia Sánchez-González, Laura Fernández-Robles, Hilde Pérez, Manuel Castejón-Limas 2019

[29]“Classification of Attacks in Wireless Sensor Networks”, Mohamed-Lamine Messai 2014

[30] “Secure Localization and Location Verification in Wireless Sensor Networks: A Survey” Yingpei Zeng · Jiannong Cao Jue Hong · Shigeng Zhang · Li Xie 2015

[31] “3D Tdoa Problem Solution with Four Receiving Nodes”, Javier Díez-González, Rubén Álvarez, Lidia Sánchez-González, Laura Fernández-Robles, Hilde Pérez and Manuel Castejón-Limas

[32] “A New Data Encryption Algorithm Based on the Location of Mobile Users” Hsien- Chou Liao and Yun -Hsiang Chao 2008

[33] “A Secure Localization Approach Using Mutual Authentication and Insider Node Validation in Wireless Sensor Networks” Gulshan Kumar, Mritunjay Kumar Rai, Hye-jin Kim and Rahul Saha 2017

[34] “A tiny hypervisor-based trusted geolocation framework with minimized TPM operations” Sungjin Parka,c, Jong-Jin Wona Jaenam Yoona, Kyong Hoon Kimb, Taisook Hanc 2016

[35] “Location Dependent Digital Rights Management” Thomas Mundt 2005.

[36] “Secure Localization Using Elliptic Curve Cryptography in Wireless Sensor Networks” V. Vijayalakshmi and Dr. T.G. Palanivelu 2008

[37] “A Collaborative Secure Localization Algorithm Based on Trust Model in Underwater Wireless Sensor Networks Guangjie Han 1,2,*, Li Liu 1,†, Jinfang Jiang 1,†, Lei Shu 3,† and Joel J.P.C. Rodrigues 2016

[38] “An Efficient Secure DV-Hop Localization for Wireless Sensor Network” Xiaole Liu, Rui Yang, Qingmin CuiPublished 2015

[39] Trusted Platform Module:] https://en.wikipedia.org/wiki/Trusted_Platform_Module

[40] Amit Vasudevan, Sagar Chaki, Limin Jia, Jonathan McCune, James Newsome and Anupam Datta “Design, Implementation and Verification of an eXtensible and Modular Hypervisor Framework”

[41] Anup Patel,MaiDaftedar,MohmadShalan ,M. Watheq El-Kharashi. Bangalore, India .Cairo, Egypt *“Embedded Hypervisor Xvisor: A comparative analysis”*

[42] Sungjin Park, Jae Nam Yoon, Cheoloh Kang, Kyong Hoon Kim and TaisookHan . "*TGVisor: A Tiny Hypervisor-Based Trusted Geo-location Framework for Mobile Cloud Clients*" . 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering

[43] Jonathan M. McCune, Ning Qu, YanlinLi ,Anupam Datta, Virgil D. Gligor, Adrian Perrig ." TrustVisor:Efficient TCB Reduction and Attestation" March 9, 2009 CMU-CyLab-09-003.

[44] "TrustVisor: Efficient TCB Reduction and Attestation" Jonathan M. McCune 2010

[45] Arvind Seshadri, Mark Luk, Ning Qu, Adrian Perrig. "SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes"CyLab/CMU Pittsburgh, PA, USA

[46] Amit Vasudevan, Bryan Parno, Ning Qu, Virgil D. Gligor and Adrian Perrig. "Lockdown: Towards a Safe and Practical Architecture for Security Applications on Commodity Platforms"CyLab/Carnegie Mellon University, Microsoft Research, Google Inc.

[47] Himanshu Raj, David Robinson, Talha Bin Tariq, Paul England, Stefan Saroiu, Alec Wolman. "Credo: Trusted Computing for Guest VMs with a Commodity Hypervisor". Microsoft Research

[48] "Experimental analysis of RSSI-based indoor localization with IEEE 802.15.4" Emanuele Goldoni, Alberto Savioli

[49] RSSI values and components https://appelsiini.net/2017/trilateration-with-n-points/

[50] "Reliable computation of the points of intersection of n spheres in IRn" I.D. Coope . 2000

[51] uberxmhfArtifact/Documentation and official webpage: https://uberxmhf.org/

[52] uberxmhfArtifact/Documentation and forum: https://forums.uberspark.org/

[53] GitHub XMHF implementation: https://github.com/anbangr/trustvisor-dev/tree/master/xmhf

[54] GitHub uberXMHF implementation: https://github.com/hypcode/uberxmhf/

[55] GitHub XVisor implementation: https://github.com/xvisor/xvisor

[56] Xvisor Artifact/Documentation and official webpage: http://xhypervisor.org/

[57] Anup Patel , Mai Daftedar,Shalan, M. Watheq El-Kharash. "Embedded Hypervisor Xvisor: A comparative analysis"

[58] TrustVistor Implementation: https://uberxmhf.org/docs/pc-legacy-x86_32/uberapp-trustvisor.html

[59] GitHub TrustVistor Implementation: https://github.com/anbangr/trustvisor-dev/tree/master/trustvisor

[60] [downgrading GRUB for Ubuntu https://ubuntuforums.org/showthread.php?t=1298932 and [downgrading GRUB for Debian https://forums.debian.net/viewtopic.php?f=17&t=50132)

[61] Install jTpmTools https://sourceforge.net/projects/trustedjava/files/jTPM%20Tools/

[62] Lockdown Implementation: https://uberxmhf.org/docs/pc-legacy-x86_32/uberapp-lockdown.html

[63] GitHub Lockdown implementation: https://github.com/anbangr/trustvisor-dev/tree/master/lockdown

[64] Simulation Source code: https://github.com/SavvasR1991/Safe-Geolocalization-Project/tree/master/Localization-Simulator

[65] Chan Location Algorithm Application in 3-Dimension Space Location, Zhang Jian-wu, Yu Cheng-lei, Tang bin, Ji Ying-ying School of Communication Engineering, Hangzhou Dianzi University. 2008

[66] “Time of Arrival (TOA)-Based Direct Location Method”, Mohamed Khalaf-Allah, Umm Al-Qura University P.O. Box 715, Makkah, 21955, SAUDI ARABIA

[67] 3D Position Estimation Performance Evaluation of a Hybrid Two Reference TOA/TDOA Multilateration System Using Minimum Configuration, Yaro Abdulmalik Shehu 2016