Design and Analysis of Access Control and User Authentication Protocols for Wireless Sensor Networks

Thesis submitted for the

**Doctor of Philosophy (Engineering)**

Degree of Jadavpur University, Kolkata, India

By

**Santanu Chatterjee**

Department of Computer Science and Engineering
Jadavpur University
Kolkata, West Bengal 700 032, India

2014
Design and Analysis of Access Control and User Authentication Protocols for Wireless Sensor Networks

by
Santanu Chatterjee

Thesis submitted for the
Doctor of Philosophy (Engineering)
Degree of Jadavpur University, Kolkata, India

Supervisors:

Dr. Ashok Kumar Das
Center for Security, Theory and Algorithmic Research
International Institute of Information Technology
Hyderabad 500 032, India

Dr. Jamuna Kanta Sing
Department of Computer Science and Engineering
Jadavpur University
Kolkata 700 032, India

2014
1. Title of the Thesis
Design and Analysis of Access Control and User Authentication Protocols for Wireless Sensor Networks

2. Name, Designation and Institution of the Supervisors:

1. Dr. Ashok Kumar Das
   Assistant Professor
   Center for Security, Theory and Algorithmic Research
   International Institute of Information Technology
   Hyderabad 500 032, India

2. Dr. Jamuna Kanta Sing
   Professor
   Department of Computer Science and Engineering
   Jadavpur University
   Kolkata 700 032, India
3. List of Publications:

1. Journal papers


2. **International conference papers**


4. **List of Patents:** Nil

5. **List of Presentations in National/International Conferences**

CERTIFICATE FROM THE SUPERVISORS

This is to certify that the thesis entitled “Design and Analysis of Access Control and User Authentication Protocols for Wireless Sensor Networks”, submitted by Sri Santanu Chatterjee, who got his name registered on 7th December, 2012 for the award of Ph.D. (Engg.) degree of Jadavpur University is absolutely based upon his own work under the supervision of Ashok Kumar Das and Jamuna Kanta Sing and that neither his thesis nor any part of the thesis has been submitted for any degree or any academic award anywhere before.

1. ________________________________________________________________
   Signature of the Supervisor
   and date with Official Seal
   Dr. Ashok Kumar Das
   Assistant Professor
   Center for Security, Theory and Algorithmic Research
   International Institute of Information Technology
   Hyderabad 500 032, India

2. ________________________________________________________________
   Signature of the Supervisor
   and date with Official Seal
   Dr. Jamuna Kanta Sing
   Professor
   Department of Computer Science and Engineering
   Jadavpur University
   Kolkata 700 032, India
Acknowledgments

I would like to acknowledge the moral and intellectual supports given to me by my supervisors Dr. Ashok Kumar Das and Dr. Jamuna Kanta Sing during my PhD program. Thanks to my supervisors’ constant guidances and approaches for which this long and difficult journey becomes smooth and interesting.

I am extremely grateful to Dr. Das. His way to do research as well as his attitude towards study and analysis of any particular subject influenced me immensely, and I still feel there is a lot to learn from him. Among other things, I have always admired his ability to discuss research problems from scratch to formalize rigorously, his scientific bravery in supporting ideas that sounded completely mental at first glance. Also his ideas and his style of writing have always impressed me and influenced my own style of work. Most important of all was probably his calm and constant belief in my ability to get a PhD.

I am very thankful to Dr. Sing. Dr. Sing helped me a lot as I started my PhD at the Jadavpur University. I thank him for his wisdom, for many fruitful discussions, moral support, and fruitful cooperation over various papers.

I also thank Prof. Sibaji Bhattacharya, Head, Department of Computer Science & Engineering, Jadavpur University for serving on my Doctoral Scrutiny Committee.

I also want to thank my Division Head, Mr. Gautam Mahapatra, Technology Director, B. N. Gajbhiye, Former Director of our lab, Dr. S. K. Chaudhuri and our present lab Director, Mr. G. Satheesh Reddy and all my Colleagues at the Research Center Imarat, Defence Research and Development Organization, Hyderabad, for their support during my entire research programme.

I would also like to thank my parents, my wife and my daughter for their moral support and patience during the course of my research work.

Finally, I would like to thank all of them whose names are not mentioned here but have helped me in any way to accomplish the work.

Research Center Imarat
Defence Research and Development Organization
Hyderabad, India

Santanu Chatterjee
Abstract

In a wireless sensor network (WSN), a large number of tiny computing nodes, called the sensors or motes, are mainly deployed in a public and uncontrolled area, known as the target field (or deployment area), for purpose of sensing the information surrounding to their areas and send back the sensed data to the nearby base stations for further processing. These sensor nodes are typically resource-constrained whereas the base station is assumed to be resource-rich. In WSN, sensors communicate among each other through short-range wireless radio communication, which is very difficult to protect as it is realized over an insecure broadcast medium. In a broadcast medium, an attacker (adversary) can easily eavesdrop on, intercept, inject, and also alter the transmitted data. Due to cost constraints, sensor nodes are not typically equipped with tamper-resistant hardware. Since wireless sensor networks are often operated in an unattended mode, an adversary may physically capture some sensors in the target field in order to compromise their all stored sensitive data including the secret keys and codes. Therefore, any adversary that gets hold of a sensor node can easily extract its stored cryptographic information. Furthermore, sensor networks are vulnerable to resource consumption attacks. Adversaries can repeatedly send packets to drain a node battery and waste network bandwidth. As a result, secure transmission of sensitive digital information over the sensor network is very much essential. For providing security in WSN, key distribution, user authentication, access control and user access control are the most essential security requirements. However, due to resource limitations and physical insecurity of sensor nodes, the design of such security protocols in WSN becomes a challenging task as the already designed security protocols for traditional networks are not appropriate for WSN applications.

In this thesis, we aim to study in the area of user authentication, access control and user access control in WSNs. In the first study, we propose a new password-based user authentication scheme in hierarchical wireless sensor networks. The proposed scheme achieves better security and efficiency as compared to those for other existing password-based approaches. In addition, the proposed scheme has merit to change dynamically the user’s password locally without the help of the base station or gateway node. The proposed scheme supports dynamic nodes addition after the initial deployment of nodes in the existing sensor network. In addition, we
analyze the proposed scheme for formal security under the random oracle models to show this scheme is secure. Furthermore, we simulate this proposed scheme for formal security verification using the widely-accepted Automated Validation of Internet Security Protocols and Applications (AVISPA) tool. AVISPA tool ensures that whether a protocol is insecure against passive and active adversaries. Using the AVISPA model checkers, we show that the proposed scheme is also secure against possible passive and active attacks, including the replay and man-in-the-middle attacks.

The second study is based on designing a new access control scheme for large-scale distributed wireless sensor networks, which is called the certificate-based access control scheme, which not only identifies the identity of each node but it has also ability to differentiate between old nodes and new nodes. The proposed scheme does not require involvement of the base station during authentication and key establishment processes, and it can be easily implemented as a dynamic access control protocol. In addition, the proposed scheme significantly reduces communication costs in order to authenticate neighbor nodes among each other and establish symmetric keys between neighbor nodes as compared with existing approaches. Further, the proposed scheme is secure against different attacks and unconditionally secure against node capture attacks, which are evident through both informal and formal security analysis. The simulation results of the proposed scheme using the AVISPA tool also ensure that our scheme is secure.

In the third study, we propose a certificate-less access control mechanism for large-scale distributed wireless sensor networks. The proposed scheme is significantly better in terms of performance and security as compared to other related access control schemes. In fact, the proposed scheme requires significantly less communication costs as compared to other related schemes. Moreover, we simulate the proposed scheme for formal security analysis using the AVISPA tool and show that our scheme is secure.

Finally, we concentrate in designing a novel user access control scheme for applications in wireless body area networks (WBANs). WBANs are envisioned to provide healthcare and patient monitoring applications. Without considering security, patient privacy is vulnerable in WBAN. Access rights for the correct information and resources for different services to the genuine users can be provided with the help of efficient user access control mechanisms. The proposed scheme makes use of the
group-based user access id, access privilege mask as well as password. The elliptic
curve cryptography-based public key cryptosystem is used to ensure that any partic-
ular legitimate user can access only those information for which he/she is permitted
to access them. We show that the proposed scheme performs better than the previ-
ously existing user access control schemes. Through the security analysis, we show
that the proposed scheme is secure against possible known attacks. Furthermore,
through the formal security verification using the AVISPA tool we show that our
scheme is also secure.
Dissemination of Work


[Chapter # 7] Santanu Chatterjee, Ashok Kumar Das, and Jamuna Kanta Sing. “A novel and efficient user access control scheme for wireless body area sensor networks,” in *Journal of King Saud University - Computer and Information Sciences (Elsevier)*, Vol. 26, No. 2, pp. 181-201, 2014 [This article is one of the top 25 most downloaded articles during February 2014 to December 2014 of the Journal of King Saud University - Computer and Information Sciences].
Contents

1 Introduction

1.1 Network models ........................................... 1
1.2 Hardware constraints .................................... 3
1.3 Sensor network topology ................................ 5
1.4 Applications .............................................. 6
1.5 General security requirements of WSN ................. 8
1.6 Sensor network limitations ............................... 10
1.7 Key distribution in WSN ................................. 11
1.8 User authentication in WSN ....................... 13
  1.8.1 Security requirements ............................. 13
  1.8.2 Functionality requirements .................... 15
1.9 Access control in WSN ................................. 16
  1.9.1 Security requirements ............................. 16
  1.9.2 Functionality requirements .................... 17
1.10 User access control in WSN ...................... 18
  1.10.1 Security requirements ............................. 18
  1.10.2 Functionality requirements .................... 19
1.11 Motivation of the work ............................... 19
1.12 Objective of the work ................................. 20
1.13 Summary of contributions ............................ 21
  1.13.1 Dynamic password-based user authentication in hierarchical
        wireless sensor networks ............................. 21
  1.13.2 Certificate-based access control in distributed wireless sensor
        networks .............................................. 22
1.13.3 Certificate-less access control in distributed wireless sensor networks ........................................ 22
1.13.4 User access control in hierarchical wireless body area sensor networks ........................................ 22
1.14 Organization of the thesis ........................................ 23

2 Mathematical Background ........................................ 25
  2.1 One-way hash function ........................................ 25
  2.2 RSA cryptosystem ........................................ 27
    2.2.1 RSA encryption/decryption ........................................ 27
    2.2.2 RSA digital signature ........................................ 29
    2.2.3 Security of RSA cryptosystem ........................................ 30
  2.3 ECC cryptosystem ........................................ 30
    2.3.1 Elliptic curve over finite field ........................................ 30
    2.3.2 ECC encryption/decryption ........................................ 33
    2.3.3 ECC signature ........................................ 35
    2.3.4 Security of ECC cryptosystem ........................................ 36
  2.4 RSA versus ECC ........................................ 38
  2.5 Summary ........................................ 38

3 Review of Related Works ........................................ 41
  3.1 Taxonomy of security protocols in WSNs ........................................ 41
  3.2 Existing user authentication schemes in WSNs ........................................ 43
  3.3 Existing access control schemes in WSNs ........................................ 46
  3.4 Existing user access control schemes in WSNs ........................................ 48
  3.5 Summary ........................................ 50

4 Dynamic Password-Based User Authentication in Hierarchical WSNs ........................................ 51
  4.1 The proposed password-based user authentication scheme ........................................ 52
    4.1.1 Network model ........................................ 52
    4.1.2 Threat model ........................................ 53
    4.1.3 Notations ........................................ 54
    4.1.4 Our contributions ........................................ 54
  4.2 Description of the proposed scheme ........................................ 56
    4.2.1 Pre-deployment phase ........................................ 57
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.2</td>
<td>Post-deployment phase</td>
<td>58</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Registration phase</td>
<td>59</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Login phase</td>
<td>61</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Authentication phase</td>
<td>63</td>
</tr>
<tr>
<td>4.2.6</td>
<td>Password change phase</td>
<td>66</td>
</tr>
<tr>
<td>4.2.7</td>
<td>Dynamic node addition phase</td>
<td>66</td>
</tr>
<tr>
<td>4.3</td>
<td>Security analysis of the proposed scheme</td>
<td>67</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Replay attack</td>
<td>67</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Many logged-in users with the same login-id attack</td>
<td>68</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Stolen-verifier attack</td>
<td>68</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Password guessing attack</td>
<td>68</td>
</tr>
<tr>
<td>4.3.5</td>
<td>Password change attack</td>
<td>69</td>
</tr>
<tr>
<td>4.3.6</td>
<td>Resilience against node capture attack</td>
<td>69</td>
</tr>
<tr>
<td>4.3.7</td>
<td>Smart card breach attack</td>
<td>71</td>
</tr>
<tr>
<td>4.3.8</td>
<td>Denial-of-service attack</td>
<td>72</td>
</tr>
<tr>
<td>4.3.9</td>
<td>Privileged-insider attack</td>
<td>72</td>
</tr>
<tr>
<td>4.3.10</td>
<td>Masquerade attack</td>
<td>73</td>
</tr>
<tr>
<td>4.3.11</td>
<td>Formal security analysis of the proposed scheme</td>
<td>73</td>
</tr>
<tr>
<td>4.4</td>
<td>Formal security verification of our scheme using AVISPA tool</td>
<td>77</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Overview of AVISPA</td>
<td>78</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Specifying the protocol</td>
<td>80</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Analysis of results</td>
<td>85</td>
</tr>
<tr>
<td>4.5</td>
<td>Comparison with related schemes</td>
<td>86</td>
</tr>
<tr>
<td>4.6</td>
<td>Summary</td>
<td>94</td>
</tr>
</tbody>
</table>

5 Certificate-Based Access Control in Distributed WSNs

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Threat model</td>
<td>96</td>
</tr>
<tr>
<td>5.2</td>
<td>Notations</td>
<td>97</td>
</tr>
<tr>
<td>5.3</td>
<td>Motivation</td>
<td>97</td>
</tr>
<tr>
<td>5.4</td>
<td>Our contributions</td>
<td>99</td>
</tr>
<tr>
<td>5.5</td>
<td>The proposed certificate-based access control scheme</td>
<td>99</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Pre-deployment phase</td>
<td>100</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Authentication and key establishment phase</td>
<td>102</td>
</tr>
<tr>
<td>5.5.3</td>
<td>Dynamic nodes addition phase</td>
<td>108</td>
</tr>
</tbody>
</table>
5.6 Analysis of the proposed scheme ........................................ 108
  5.6.1 Functionality analysis ............................................ 108
  5.6.2 Security analysis .................................................. 111
5.7 Formal security verification of our scheme using AVISPA tool ...... 114
  5.7.1 Specifying the protocol .......................................... 115
  5.7.2 Analysis of results ................................................ 119
5.8 Performance comparison with related schemes ......................... 120
5.9 Summary ..................................................................... 124

6 Certificate-Less Access Control in Distributed WSNs ............... 127
  6.1 Notation .................................................................. 128
  6.2 Review of Huang’s access control scheme ......................... 129
    6.2.1 Pre-deployment phase .......................................... 129
    6.2.2 Authentication and key establishment phase ............... 129
    6.2.3 Dynamic node addition phase .................................. 130
  6.3 An attack on Huang’s access control scheme ....................... 131
  6.4 The proposed certificate-less access control scheme ............. 134
    6.4.1 Threat model ...................................................... 135
    6.4.2 Different phases .................................................. 135
  6.5 Analysis of the proposed scheme .................................... 141
    6.5.1 Overheads ......................................................... 142
    6.5.2 Security analysis .................................................. 143
  6.6 Formal security verification of our scheme using AVISPA tool ... 150
    6.6.1 Specifying our scheme .......................................... 151
    6.6.2 Analysis of results ................................................ 154
  6.7 Performance comparison with related schemes ...................... 154
  6.8 Summary .................................................................. 158

7 User Access Control in Hierarchical WBANs ......................... 159
  7.1 Motivation .................................................................. 161
  7.2 Threat model ........................................................... 161
  7.3 Our contributions ...................................................... 162
  7.4 The proposed user access control scheme ......................... 163
    7.4.1 Notations ........................................................... 164
    7.4.2 Different phases ................................................... 164
## Contents

### 7.5 Analysis of the proposed scheme

7.5.1 Computational overhead . . . . . . . . . . . . . . . . . . . . . 175
7.5.2 Communication overhead . . . . . . . . . . . . . . . . . . . . . 175
7.5.3 Storage overhead . . . . . . . . . . . . . . . . . . . . . . . . . . 176
7.5.4 Network scalability . . . . . . . . . . . . . . . . . . . . . . . . 176
7.5.5 Security analysis . . . . . . . . . . . . . . . . . . . . . . . . . . 177

### 7.6 Formal security verification of our scheme using AVISPA tool

7.6.1 Specifying our scheme . . . . . . . . . . . . . . . . . . . . . . 186
7.6.2 Analysis of results . . . . . . . . . . . . . . . . . . . . . . . . . 189

### 7.7 Performance comparison with other related schemes

7.7.1 Comparison of computational costs . . . . . . . . . . . . . . . 192
7.7.2 Comparison of communication costs . . . . . . . . . . . . . . . 193
7.7.3 Comparison of energy costs . . . . . . . . . . . . . . . . . . . . 194
7.7.4 Comparison of functionality . . . . . . . . . . . . . . . . . . . . 195

### 7.8 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 197

### 8 Conclusion and Future Works

8.1 Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 199
8.2 Future research directions . . . . . . . . . . . . . . . . . . . . . . . . 201
List of Figures

1.1 A hierarchical wireless sensor network (HWSN) architecture (Source: [24], [33], [36], [41], [55]). ......................................................... 2
1.2 A distributed wireless sensor network (DWSN) architecture (Source: [9], [33]). ................................................................. 4
1.3 The hardware components of a sensor node (Source: [33], [103]). . . . 4
1.4 A general three-tier architecture of WBAN (Source: [99]). ............... 7
1.5 A hierarchical WBAN with the clustering heads (Source: [22]). ....... 9

2.1 Example of elliptic curve in case of $y^2 = x^3 + x + 1 \pmod{23}$ (Source: [26]). ................................................................. 32

3.1 Security protocols in wireless sensor networks: a taxonomy. ........... 42

4.1 Security measures at gateway in wireless body area networks (Source: [11], [44]). ................................................................. 52
4.2 A hierarchical wireless sensor network (HWSN) architecture. ......... 53
4.3 Architecture of the AVISPA tool v.1.1 (Source: [3], [112]). ............ 79
4.4 Role specification in HLPSL for the user $U_i$. ............................. 81
4.5 Role specification in HLPSL for the base station, BS. ................. 83
4.6 Role specification in HLPSL for the cluster head $CH_j$. ............... 84
4.7 Role specification in HLPSL for the environment and goal. .......... 85
4.8 Role specification in HLPSL for the session. .............................. 86
4.9 The result of the analysis using OFMC back-end. ....................... 87
4.10 The result of the analysis using CL-AtSe back-end. .................... 87

5.1 Role specification for the initiator (sensor node $u$) for our protocol. . 116
5.2 Role specification for the responder (sensor node $v$) for our protocol. . 117
# List of Algorithms

1. $\text{EXP}^{\text{DLP}}_{G, g}(A)$ ................................................. 37
2. $\text{Exp}^{\text{IND-CPA}}_{\text{DPUS}_A, A}$ ........................................... 75
3. $\text{Exp}^{\text{IND-CPA, HASH}}_{\text{DPUS}_A} ............................................ 76$
4. $\text{Experiment}^{\text{ECDLP}}_{\text{ACP}, A}$ ............................................. 113
5. $\text{Experiment}^{\text{ECDLP}}_{\text{EACP}, A}$ ............................................. 147
6. $\text{Exp}^{\text{HASH, IND-CPA}}_{\text{UACS}_A}$ ........................................... 183
7. $\text{Exp}^{\text{HASH, IND-CPA, ECDLP}}_{\text{UACS}_A}$ .................................. 184
List of Tables

1.1 Basic characteristics of typical MICA2 and MICA2-DOT motes (Source: [33], [103]). ................................................................. 5

2.1 Points over the elliptic curve $E_{23}(1, 1)$ (Source: [26]). ............ 32

4.1 Notations used in this chapter. .................................................. 55
4.2 Registration phase of our proposed scheme. .............................. 61
4.3 Login phase of our proposed scheme. ....................................... 62
4.4 Authentication phase of our proposed scheme. .......................... 65
4.5 Simulation results of formal security verification among different schemes using AVISPA tool. ..................................................... 88
4.6 Performance comparison between the proposed scheme and other related schemes. ................................................................. 89
4.7 Comparison of computational costs in different phases between the proposed scheme and the other schemes. ................................. 91
4.8 Comparison of communication costs between the proposed scheme and the other schemes. ......................................................... 92
4.9 Comparison of energy cost of sensor node/cluster head between the proposed scheme and the other schemes. ................................. 93

5.1 Notations used in the proposed scheme. ..................................... 97
5.2 Certificate of a node $u$. ............................................................. 101
5.3 Authentication and key establishment phase of our proposed scheme. 106
5.4 Bit size of the parameters used for our proposed access control scheme.110
5.5 Message size in bits and number of packets needed to transmit during authentication and key establishment phase of our proposed access control scheme. ......................................................... 110
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6</td>
<td>Time complexity of various operations in terms of $T_{\text{mul}}$.</td>
<td>121</td>
</tr>
<tr>
<td>5.7</td>
<td>Comparison of computational overheads among different access control schemes.</td>
<td>121</td>
</tr>
<tr>
<td>5.8</td>
<td>Comparison of communication costs among different access control schemes.</td>
<td>122</td>
</tr>
<tr>
<td>5.9</td>
<td>Comparison of storage overheads among different access control schemes.</td>
<td>123</td>
</tr>
<tr>
<td>5.10</td>
<td>Summary of the results of formal security verification using OFMC and AtSe model checkers for our scheme and existing schemes.</td>
<td>123</td>
</tr>
<tr>
<td>5.11</td>
<td>Comparison of energy cost of a sensor during authentication and key establishment phase between our scheme and other schemes.</td>
<td>125</td>
</tr>
<tr>
<td>6.1</td>
<td>Notations used in the proposed scheme.</td>
<td>128</td>
</tr>
<tr>
<td>6.2</td>
<td>Authentication and key establishment phase of Huang’s scheme [78].</td>
<td>130</td>
</tr>
<tr>
<td>6.3</td>
<td>An active attack on Huang’s scheme during the authentication and key establishment phase.</td>
<td>132</td>
</tr>
<tr>
<td>6.4</td>
<td>Authentication and key establishment phase of our scheme.</td>
<td>140</td>
</tr>
<tr>
<td>6.5</td>
<td>Notations used for analysis of computational overhead.</td>
<td>143</td>
</tr>
<tr>
<td>6.6</td>
<td>Bit size of the parameters used for our proposed access control scheme.</td>
<td>144</td>
</tr>
<tr>
<td>6.7</td>
<td>Performance comparison of our access control scheme with other schemes.</td>
<td>157</td>
</tr>
<tr>
<td>6.8</td>
<td>Summary of the results of formal security verification using OFMC and CL-AtSe backends for our scheme and existing schemes.</td>
<td>158</td>
</tr>
<tr>
<td>7.1</td>
<td>Notations used in the proposed scheme.</td>
<td>165</td>
</tr>
<tr>
<td>7.2</td>
<td>The registration phase of our proposed scheme.</td>
<td>168</td>
</tr>
<tr>
<td>7.3</td>
<td>The login phase of our proposed scheme.</td>
<td>171</td>
</tr>
<tr>
<td>7.4</td>
<td>Authentication phase of our proposed scheme.</td>
<td>173</td>
</tr>
<tr>
<td>7.5</td>
<td>Size (in bits) of different parameters used for our scheme.</td>
<td>175</td>
</tr>
<tr>
<td>7.6</td>
<td>Message size and number of packets to be transmitted per message for our scheme during the login and authentication phases.</td>
<td>176</td>
</tr>
<tr>
<td>7.7</td>
<td>Time complexity of various operations in terms of $t_{\text{mul}}$.</td>
<td>192</td>
</tr>
<tr>
<td>7.8</td>
<td>Comparison of computational costs for different phases in different access control schemes.</td>
<td>193</td>
</tr>
<tr>
<td>7.9</td>
<td>Comparison of communication costs between the proposed scheme and the other schemes.</td>
<td>194</td>
</tr>
</tbody>
</table>
7.10 Comparison of energy cost of a sensor node during the login and authentication phases between our scheme and other schemes. 195

7.11 Comparison of functionality analysis between the proposed scheme and the other schemes. 196

7.12 Summary of the results of formal security verification using OFMC and CL-AtSe backends for our scheme and existing schemes. 197
Chapter 1
Introduction

In a wireless sensor network (WSN), a large number of small computing nodes, called sensors or motes, are scattered in an area (called the deployment field or target field) for the purpose of sensing important information and transmitting those sensing information to the nearby base stations for further processing. Sensor nodes are generally deployed densely in a close proximity to the phenomenon to be monitored. A sensor node is a node in a WSN that is capable of performing some processing, gathering sensory information and communicating with other connected sensor nodes in that network. Sensor nodes communicate among each other by short range radio communications. The base station is a computationally well-equipped node in the network, whereas the sensor nodes are resource-starved. The sensor nodes are usually scattered in a sensor field (i.e., deployment area or target field) and each of the scattered nodes has also the capabilities to collect data and route data back to the base station via a multi-hop infrastructure-less communication through other sensor nodes.

1.1 Network models

There are two types of WSN architectures available for wireless sensor networks: first one is the hierarchical architecture and the other is the distributed (homogeneous) architecture.

In a hierarchical wireless sensor network (HWSN) shown in Figure 1.1, there is a hierarchy among the nodes based on their capabilities: base stations, cluster heads and sensor nodes [24], [33], [36], [41], [55]. Sensor nodes are inexpensive, have
limited capability and are generic wireless devices. Each sensor has limited battery power, memory size and data processing capability and short radio transmission range. Sensor nodes in a cluster (group) communicate among each other in that cluster and finally communicate with the cluster head. Cluster heads are more resource-rich than sensors. They may be equipped with high power batteries, larger memory storage, powerful antenna and data processing capabilities, and they can execute relatively complicated numerical operations than sensors and have much larger radio transmission range. Cluster heads can communicate with each other directly and relay data between its cluster members and the base station. A cluster head can be a PDA (personal digital assistant) [10], [24], or an IMote2 [22]. Finally, a base station (also called the sink node or the gateway node (GW-node) [58]) is typically a gateway to another network, which is treated as a powerful data processing/storage center, or an access point for human interface. The base station then collects sensor readings, performs costly operations on behalf of sensor nodes and manages the network. In most applications, the base station is assumed to be trusted. As a result, the base station can also be used as key distribution center or key setup server for the key management problem in WSNs [24], [33], [36], [41], [55]. Furthermore, a sink node or base station can act as a central or certificate authority depending on the applications used in the thesis [58], [144], [151]. In addition, for the healthcare applications, the BS can act as a medical server [10]. Sensor nodes are deployed around one or more hop neighborhood of the base station.
1.2 Hardware constraints

Since the cluster heads are more powerful nodes than sensor nodes, a cluster head can also directly communicate with the BS or indirectly via its neighbor cluster heads in WSN. As pointed out in [24], the sink node (the base station) is the most powerful node in a wireless sensor network, it has virtually unlimited computational and communication power, unlimited memory storage capacity, and very large radio transmission range which can reach all the nodes in a network. Depending on the applications, the base station can be located either in the center or at a corner of the network. Since the sensor nodes have limited radio transmission ranges (typically, 100 feet or 30 meters [33]), if they want to communicate with the base station, they need to contact with their neighbor sensor nodes and the cluster heads. In this case, we then require a multi-hop communication path between a sensor node and the base station. The data flow in such networks are of three types:

- pairwise (unicast) among sensor nodes,
- group-wise (multicast) within a cluster of sensor nodes, and
- network-wise (broadcast) from base station to sensor nodes.

On the other hand, in a distributed wireless sensor network (DWSN) shown in Figure 1.2, there is no fixed infrastructure and network topology is not known prior to deployment of the sensor nodes in the target field. The sensor nodes are usually deployed all over the target area randomly and after deployment sensor nodes form an infrastructure-less multi-hop wireless communication between them and data is routed back to the base station. The data flow in DWSN is similar to data flow in HWSN with a difference that network-wise (broadcast) flow takes place for every sensor node in the network. The advantage of using HWSN over DWSN is that the sensing data by the sensor nodes could be reached to the BS via other sensor nodes or cluster heads in a few number of hops as compared to that for DWSN [24]. More details on survey on sensor networks can be found in [9].

1.2 Hardware constraints

A sensor node consists of four basic components: (i) a sensing unit, (ii) a processing unit, (iii) a transceiver unit, and (iv) a power unit, as shown in Figure 1.3. Sensor node may also have additional components like external memory, a location finding
system, power generator, and mobilizer. Sensing units are basically of two types: (i) passive sensors like camera and (ii) active sensors like radar. Sensing units are usually composed of two subunits: sensor and ADC (analog to digital converter). After sensing, the analog data are converted by ADC to digital data, and then fed
1.3 Sensor network topology

to the processing unit for further processing. Transmission unit helps to connect the sensor node to the network. Power unit is another important component of a sensor node, which is in general battery powered. In most of the sensor networks, routing techniques and sensing tasks require knowledge of location with high accuracy. Thus, it is common that a sensor node will have a location finding system. A mobilizer may sometimes be required in order to move a sensor node when it is required to carry out the assigned tasks.

The basic characteristics of typical MICA2 and MICA2-DOT motes [6] are shown in Table 1.1. MICA2 and MICA2-DOT motes are widely used in current generation of sensor networks. Note that sensor nodes are extremely resource constrained. A sensor node contains a primitive processor featuring very low computing speed and only a small amount of programming memory. Since a sensor node is battery powered, so the energy budget is a major concern in wireless sensor networks while designing a security protocol.

Table 1.1: Basic characteristics of typical MICA2 and MICA2-DOT motes (Source: [33], [103]).

<table>
<thead>
<tr>
<th></th>
<th>MICA2</th>
<th>MICA2-DOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>8-bit 7.7 MHz Atmega 128</td>
<td>8-bit 4 MHz Atmega 128</td>
</tr>
<tr>
<td>RAM</td>
<td>4K bytes</td>
<td>4K bytes</td>
</tr>
<tr>
<td>ROM</td>
<td>128K bytes</td>
<td>128K bytes</td>
</tr>
<tr>
<td>EEPROM</td>
<td>512K bytes</td>
<td>512K bytes</td>
</tr>
<tr>
<td>Data rate</td>
<td>38.4K baud</td>
<td>38.4K baud</td>
</tr>
<tr>
<td>Default packet size (under TinyOS [75])</td>
<td>29 bytes</td>
<td>29 bytes</td>
</tr>
<tr>
<td>Power supply</td>
<td>2 AA batteries</td>
<td>1 coin cell battery</td>
</tr>
</tbody>
</table>

1.3 Sensor network topology

The lifetime of wireless sensor network depends on three phases: (i) pre-deployment phase, (ii) post-deployment phase, and (iii) redeployment of additional nodes phase. Topology of WSN is dynamic in nature and may change phase-wise. Sensor nodes
may expire due to battery-energy consumption and also new sensor nodes may be needed to deploy to the network in order to replace battery-exhausted nodes and malicious nodes. The details of three phases are given below:

**Pre-deployment phase:** The sensor nodes may be deployed all over the target area using one of the following methods:

- randomly by throwing sensor nodes from an airplane or truck.
- through an artillery shell, rocket or missile.
- planned way like grid-based deployment by human or robot.

**Post-deployment phase:** After the deployment, the topology of the sensor network may change because of the following problems:

- coverage problems of sensor nodes due to jamming, noise, etc.
- irregularities in the sensor field like obstacles.
- sensor node’s battery-energy constraints.
- sensor node’s malfunctioning.

**Redeployment of additional nodes phase:** Redeployment and replacement of sensor nodes are inevitable due to the following reasons:

- sensor node can be physically captured or compromised by an attacker from the target field.
- due to battery-energy constraints, some nodes can expire.

### 1.4 Applications

Sensor networks are widely deployed in a variety of applications ranging from military to environmental and medical research. In many applications, such as target tracking, battlefield surveillance and intruder detection, WSNs often operate in hostile and unattended environments. Therefore, there is a strong need for protecting the sensing data and sensing readings. In wireless environments, an adversary not only can eavesdrop the radio traffic, but also has the ability to intercept or interrupt the exchanged messages. Thus, many protocols and algorithms do not work in
1.4 Applications

hostile environments without adequate security measures. Hence, security becomes one of the major concerns when there are potential attacks against sensor networks.

Consider the scenario of battlefield surveillance which is one of the major military applications. A large number of sensor nodes are rapidly deployed in a battlefield via airplanes or trucks. Each individual sensor node monitors conditions and activities in its surroundings after deployment in the battlefield and then reports these sensing observations to the base station via wireless communications through its neighboring sensor nodes. The base station then can conduct a more accurate detection on the activities (for example, possible attacks) of the opposing force after collecting a large number of sensing observations from the sensor nodes. Thus, the appropriate decisions as well as responses can be made quickly in the battlefield.

Figure 1.4: A general three-tier architecture of WBAN (Source: [99]).

Consider another application of WSNs in the field of healthcare medical research. In a wireless body area sensor network (WBAN), the low-power tiny sensor nodes are placed around a patient’s body for monitoring the patient’s body functions and neighboring environment of a patient [70], [101], [119], [155]. A typical example of WBAN is shown in Figure 1.4 [99]. With the help of WBAN, the patient’s health
related information, such as temperature, respiration, heart rate, pulse oximeter, blood pressure, blood sugar, pH, etc. can be monitored remotely [11]. These health related information must be continuously processed in real time. The medical information need to be shared and accessed by various levels of users, such as healthcare staff, researchers, government agencies, and insurance companies for taking the important decisions such as clinical diagnosis and emergency medical responses about the patients [99]. The bio-sensors are placed in a patient’s body in order to transmit sensing data in a secure channel to a small body area network gateway. The gateway then processes data locally and resends it through a secure channel to the router for the external network to the medical server at the hospital. The results are then observed and analyzed by medical staffs/doctors to monitor patients. In this scenario, a patient wears various bio-sensors. A centralized control device is used for data transmission from in and out of the network. This control device can also be used as the gateway between internal network and base station. The base station is connected with the external network.

We have also considered another hierarchical WBAN as shown in Figure 1.5 [22], which is a two-tier hierarchical architecture. In this architecture, the cluster heads collect, aggregate and fuse the data from other sensor nodes in that cluster (body area network (BAN)), and then check data consistency and transmit the processed data to the base station or medical server. In this architecture, the high level tier is used for the inter-BAN communication, whereas the low level tier is used for the intra-BAN communication. Even for the inter-BAN communication, BANs are sometimes scattered and the base station (medical server) is not always in their ranges so that a BAN must cooperate with one another BAN using hop by hop communication through the cluster heads in order to reach the base station (medical server).

1.5 General security requirements of WSN

Security requirements in WSNs are very similar to those of ad-hoc networks. WSNs have the following general security requirements [37]:

- Authentication: Authenticating other sensor nodes, cluster heads, and base stations before granting a limited resource, or revealing information
1.5 General security requirements of WSN

Figure 1.5: A hierarchical WBAN with the clustering heads (Source: [22]).

- **Integrity**: Ensuring that the message or the entity under consideration must not be altered.

- **Confidentiality**: Providing privacy of the wireless communication channels in order to prevent false reports injection.

- **Availability**: Ensures that the desired network services are available even in the presence of denial of service attacks.

- **Non-repudiation**: Preventing malicious nodes to hide their activities.

- **Authorization**: Ensures that only the sensor nodes those who are authorized can be involved in providing information to network services.

- **Freshness**: Ensures that the data is recent and no adversary can replay old messages.
Apart from these security requirements, the forward and backward secrecy need to be considered as new sensors can be deployed in the network and old sensors may fail due to energy problems.

- **Forward secrecy:** When a sensor node leaves the network, it must not read any future messages after its departure.

- **Backward secrecy:** When a new deployed node joins in the network, it must not read any previously transmitted message.

### 1.6 Sensor network limitations

However, the following features of sensor networks make it particularly challenging task to provide the above security requirements for a sensor network [37], listed in Section 1.5:

- **Limited resources in sensor nodes:** Each sensor node has a primitive processor featuring very low computing power and only small amount of programmable memory.

- **Limited life-time of sensor nodes:** Each sensor node is battery-powered. So, after several weeks or months of operation, some nodes in the network may exhaust their power and as a result, the security protocols used must be energy efficient.

- **Limited communication abilities of sensor nodes:** Sensor nodes have the ability to communicate with each other and the base stations using short range wireless radio transmission at low bandwidth.

- **Lack of knowledge about deployment configuration:** In most applications, the post-deployment network configuration is not possible to decide a prior. As a result, it may not be always possible to use security algorithms that have strong dependence on locations of sensor nodes in a sensor network.

- **Issue of node capture:** Wireless sensor networks often operate in an unattended environment. An adversary may physically capture some sensors to compromise their stored sensitive secret data and codes from their memory as they are not generally equipped with tamper-resistant hardware.
1.7 Key distribution in WSN

To provide the above security requirements, listed in Section 1.5, the key pre-distribution method had been popularly used in the literature. In this method, the practical approach is to preload a set of keying information before the deployment of sensor nodes in the target field. After deployment, they discover their neighbor nodes and then establish the secret keys between them using the preloaded keying information. The simplest solution is deterministic approach which uses a single mission master key for the entire network. In this case, in the key pre-distribution phase each node is given the same mission key before deployment in the network. After deployment, in the key establishment phase any two neighbor nodes can communicate securely with each other using that key. However, the main drawback of this simple approach is that the compromise of even a single node in a network would reveal the secret key and thus allow decryption of all network traffic. Another solution of this approach is to use a single shared network-wide key to establish session keys between any two neighbor nodes during the key establishment phase, and then erase the network-wide key. However, the main difficulty of such a variant of the key establishment procedure is that it does not allow deployment of new nodes after the initial deployment in the network.

Other way to provide secure communication with the help of random key pre-distribution approach. Eschenauer and Gligor in 2002 first proposed a seminal random key pre-distribution scheme [66], which consists of the following three phases:

- In the key pre-distribution phase, the (key) setup server (usually the base station) selects a large key pool consisting of randomly generated symmetric keys. Each key is assigned a unique identifier in the key pool. The setup server then chooses a random subset of smaller size from the pool, called the key ring and loads this key ring into its memory before its deployment.

- In direct key establishment phase (also called the shared key discovery phase), each sensor node locates all its physical neighbors within its communication range. In order to establish a secret pairwise key between two neighbor nodes, they exchange the key ids from their key rings. If there is a common key id between their key rings, the corresponding key is taken as the secret key between them. Later, they use this established key for their future secure communication. Nodes which discover that they have a shared secret key in
their key rings then verify that their neighbor actually holds the key through a challenge-response protocol.

- The *path key establishment phase* is an optional phase, and if executed, adds to the network connectivity of the network. Suppose two neighbor nodes $u$ and $v$ fail to establish a secret key between them during the direct key establishment phase, but there exists a secure path between them. Once such a secure path is discovered, $u$ generates a new random key $k$ and securely transmits it along this path to the desired destination node $v$. In this way, $u$ and $v$ can communicate secretly and directly using $k$. However, the main problem is that the communication overhead increases significantly with the number $h$ of hops. For this reason, in practice, $h$ is restricted to a small value. When the key pool size is chosen smaller, this scheme provides high network connectivity, that is, any two neighbor nodes can establish a secret key using their key rings with high probability. On the other hand, if some nodes are compromised by an attacker, the probability of compromising a secure link between any two neighbor non-compromised nodes is also high since the key pool size is smaller, and as a result the resilience against node capture becomes poor. Some improved alternatives to the path key establishment have been proposed in the literature [28], [35], [49], [146]. They provide better trade-offs between overheads, network connectivity and resilience against node capture as compared to those for the path key establishment.

After that several improvements on the basic random key distribution scheme have been proposed in the literature, some of them are [19], [41], [64], [104]. Numerous symmetric key pre-distribution and authentication protocols have been proposed to protect sensor networks [19], [29], [30], [31], [32], [34], [38], [40], [41], [51], [56], [54], [64], [65], [66], [104] (see surveys [33], [37], [140], [148] for details). These protocols can establish symmetric pairwise secret keys between neighbor nodes in the sensor network with simple computations and they can reduce the risk of entire sensor network. However, most of the protocols can not be easily implemented as a dynamic access control because all the existing old keys as well as broadcasting messages of existing nodes should be updated once new nodes are deployed in the network [77], [78].
1.8 User authentication in WSN

In a user authentication in WSN, a legitimate user is allowed to query and collect the real-time data at any time from a sensor node or cluster head of the network as and when he/she demands for it. As most of the applications in wireless sensor network (WSN) are real-time based, so users are generally interested in accessing real-time information. This is possible if the users (called the external parties) are allowed to access the real-time data directly from the nodes inside WSN and not from the base station $BS$. Usually, the information from nodes are gathered periodically in the $BS$ and so, the gathered information may not be real-time. In order to get the real-time information from the nodes, the user needs to be first authorized to the nodes as well as the $BS$ so that illegal access to nodes do not happen. As a result, the user authentication problem becomes a very important topic in research of WSN security.

User authentication should be offered to protect the sensing information and also to prevent access of the illegal users (attackers). We list the security and functionality requirements needed for an idle user authentication scheme in WSN in the following subsections.

1.8.1 Security requirements

According to the security requirements for user authentication in WSNs, the following attacks must be prevented:

- **Replay and man-in-the-middle attacks** A replay attack is an offensive action in which an attacker tries to deceive another legitimate user in the network through the reuse of information obtained in a protocol. Thus, this attack indicates an attempt by an unauthorized third party to record the exchanged messages during transmissions. In a man-in-the-middle attack, an attacker may intercept the messages during transmissions and can change/delete/modify the content of the messages delivered to the recipients. These type of attacks should be protected by a user authentication scheme.

- **Many logged-in users with the same login-id attack** Those systems which maintain the password/verifier table to verify user login are usually vulnerable to many logged-in users with the same login-id attack. If more
than one legitimate user have same login-id and password, any of those users can launch this attack.

- **Stolen-verifier attack** This type of attack happens when the BS stores any verifier/password table for user verification. The attacker can steal any user’s login id or password from the table. It is also noted that the BS, cluster heads and sensor nodes should not keep password tables to resist against this kind of attack. Hence, one of the interesting characteristics of a user authentication scheme is that it does not require to store any verifier/password table for verification.

- **Password guessing attack** In this attack, an attacker tries to guess the password of a genuine user either online or offline through the transmitted messages and some stored secret information in the system. In a user authentication scheme, such attack should be protected.

- **Password change attack** In this attack, an attacker tries to change the password of a legal user. For example, if the smart card of the legal user in a smart card based user authentication scheme is compromised, the attacker can breach the information stored in the smart card to change the password of that user.

- **Resilience against node capture attack** The resilience against node capture attack of a user authentication scheme in WSN is measured by estimating the fraction of total secure communications that are compromised by a capture of $c$ nodes not including the communication in which the compromised nodes are directly involved. In other words, we want to find out the effect of $c$ sensor nodes being compromised on the rest of the network. For example, for any non-compromised sensor node $S_j$, we need to find out what is the probability that the adversary can decrypt the secure communication between the sensor node $S_j$ and a user $U_i$ when $c$ sensor nodes are already compromised? Let this probability be denoted by $P_e(c)$. If $P_e(c) = 0$, we call such a scheme is perfectly secure or unconditionally secure against node capture attack. A user authentication scheme should be highly resilient against node capture attack.

- **Smart card breach attack** Although the smart card is assumed safe and cannot be cracked, however there is a risk of smart card crack. If an at-
tacker/intruder attains a smart card and cracks it, he/she can obtain cryptographic information stored in it by launching the power analysis attacks [111]. Any ideal user authentication scheme should be designed in such a way that even if the smart card is stolen and cracked, the attacker should not be able to know the user’s secret credentials.

- **Denial-of-service attack** A denial-of-service (DoS) attack is any event that diminishes or eliminates a network’s capacity to perform its expected function. Hardware failures, software bugs, resource exhaustion, environmental conditions, or any complicated interaction between these factors can cause a DoS [145]. A user authentication scheme should prevent this attack.

- **Privileged-insider attack** In this kind of attack the system manager or a privileged-insider of the BS attempts to know the login details of any genuine user. In user authentication scheme, it should be taken care that the login details of any user should not be compromised by any privileged-insider or system manager.

- **Masquerade attack** In masquerade attack, an illegal user may try to fabricate a fake login request message to cheat the BS to convince that it is a legal login request in the login phase. A user authentication scheme should prevent this kind of attack.

### 1.8.2 Functionality requirements

The basic functionality requirements for user authentication problems are listed below.

- An authorized user should change his/her password freely and completely locally without the help of the BS at any time for security reasons.

- New nodes can be dynamically added into the existing sensor nodes after initial deployment.

- A user authentication protocol should be designed in such a way that it requires the minimum number of message/packet transmissions during the login and authentication phases for resource constraint nature of sensor nodes. In
addition, it should be also computationally efficient and the storage requirement in each sensor node must be minimum.

- In user authentication protocol, the scalability should be provided with respect to user and sensor node. It must support a large number of nodes in the network.

1.9 Access control in WSN

An access control scheme consists of two tasks: node authentication and key establishment. In node authentication, a deployed node needs to prove its identity to its neighbor nodes and also to prove that it has the right to access the existing sensor network. On the other hand, in key establishment, the secret shared keys need to be established between a deployed node and its neighbor nodes to protect secure communications among them. According to the previous researches [77], [85], [151], we list some essential requirements for evaluating an access control scheme designed for wireless sensor networks as follows.

1.9.1 Security requirements

According to the security requirements for access control in WSNs, the following attacks must be prevented:

- **Withstand external devices to eavesdrop or inject data:** An attacker may try to eavesdrop or inject false reports into the sensor networks. An access control protocol must prevent external devices from such eavesdropping or injecting reports into the existing sensor networks.

- **Resilience against node capture attacks:** The resilience against node capture attack of an access control scheme is measured by estimating the fraction of total secure communications that are compromised by a capture of \(c\) sensor nodes *not including* the communication in which the compromised nodes are directly involved. In other words, we wish to find out the effect of \(c\) sensor nodes being compromised on the rest of the network. For example, for any two non-compromised sensor nodes \(u\) and \(v\), we need to find out the probability that the adversary can decrypt the secret communications between \(u\) and \(v\)
when $c$ sensor nodes are already compromised. An access control scheme must be highly resilient against node capture attacks.

- **Resilience against new node deployment attacks:** An access control scheme must defend against malicious node deployment attack, Sybil attack, node replication attack and wormhole attack. In the Sybil attack [63], [114], a malicious node illegitimately takes on multiple identities. Thus, the impersonated identities may belong to existing nodes or non-existing nodes. These malicious nodes may be deployed directly by an adversary or they could be compromised nodes in the network. Such kind of attack may pose a very serious threat to distributed storage, routing protocols, data aggregation, voting, fair resource allocation, misbehavior detection, etc. Wormhole attack is an attack [76], where an adversary can tunnel messages received in one part of the network over a low latency link and replay them in a different part of the network. This attack may distort the network topology by making two distant nodes believe that they are neighbors and hence, it becomes a very serious threat to routing protocols. In node replication attack [120], an adversary can intentionally put many replicas of a compromised node at many places in the network in order to incur inconsistency. Like the Sybil attack, this attack can also render adversary the ability to subvert data aggression, misbehavior detection and voting protocols by injecting false data or suppressing legitimate data. Thus, in this attack an adversary can capture a set of sensor nodes in the network and then fabricate many replicas of those nodes with the information gathered from those captured nodes, and then place these replicas back into the strategic positions in the network for further malicious activities.

### 1.9.2 Functionality requirements

The basic functionality requirements for access control problems are listed below.

- An access control scheme should support dynamic nodes addition into the existing sensor network after initial deployment of nodes. This is required due to the loss of power to the sensor nodes after several weeks or months of operation. Further, some nodes could be compromised in the network by the attacker. Thus, new nodes are to deployed to extend the lifetime of the network.
• An access control scheme must provide mutual authentication between any two neighbor sensor nodes for pairwise key establishment.

• An access control scheme should provide very high secure connectivity in the network, that is, any two neighbor nodes should be able to establish secret pairwise key between them.

• An access control scheme should be designed in such a way that it requires minimum number of message/packet transmissions during the authentication and key establishment phase in order to use it for practical applications. In addition, it should be computationally efficient, and the storage requirement in each sensor node must be minimum.

• An access control scheme should not involve the base station (central authority) during the authentication and key establishment phase, and dynamic node addition phase to avoid extra communication and computational overheads. Further, an access control scheme should allow any two neighbor nodes to authenticate and establish secret keys between them locally without involving the base station. Thus, no matter how many nodes are deployed in a sensor network, the communication and computational overheads should remain minimum due to only authentication and establishment of secret keys between their neighbor nodes. As a result, the designed access control scheme needs to be scalable, that is, it should support a large-scale network.

1.10 User access control in WSN

User access control mechanism provides the access rights for the correct information and resources for different services in wireless sensor network. Using user access control, an authorized user can access only those information for which he/she is permitted to access. We list the following security and functionality requirements for a user access control scheme:

1.10.1 Security requirements

A user access control scheme should have the ability to defend various known attacks such as stolen-verifier attack, many logged-in users with the same login-id attack,
resilience against node capture attack, masquerade attack, replay attack, privileged-insider attack, smart card breach attack and denial-of-service attack.

1.10.2 Functionality requirements

The basic functionality requirements for user access control problems are as follows:

- Scalability should be provided, that is, the size of the network must be flexible against substantial increase of sensor nodes.

- Allow an authorized user to change his/her password dynamically at any time without involvement of the base station.

- User access control should take minimum number of message/packet transmissions during the authentication and login phase. In addition, it should be also computationally efficient and the storage requirement in each sensor node must be minimum.

1.11 Motivation of the work

Limited resource, unattended operation and wireless communication medium make wireless sensor network infeasible to use the security protocols designed for the traditional networks. Sensor nodes are not traditional computing devices, and as a result, existing security models and methods are not much suited on this environment. However, sensor networks also share many characteristics with traditional network security requirements and attack models. Furthermore, a WSN is vulnerable to various kinds of attacks, which are listed in Sections 1.8.1, 1.9.1 and 1.10.1. To prevent those attacks, different types of security protocols such as key distribution, user authentication, access control and user access control, are extremely needed in WSN.

To provide the security requirements in WSN, several key distribution schemes have been already proposed in the literature. However, most of these schemes can not be easily implemented as dynamic access control schemes. The problems of user authentication, access control and user access control have drawn little attention in the research of wireless sensor network security. User authentication in wireless sensor networks is a critical security issue due to their unattended and hostile
deployment in the field. For critical applications of WSNs there is a great need to access the real time data inside the WSN from the nodes, because the real-time data may no longer be accessed through the base station only. So, the real-time data can be given access directly to the external users (parties) those who are authorized to access data as and when they demand. Thus, user authentication plays a vital role for the problem of protecting sensor network data from illegitimate access. New node deployment in sensor networks is necessary requirement due to the loss of sensor nodes because after several weeks or months of operation some sensor nodes in the network may exhaust their power. A new deployment node may not always be a legitimate node. Moreover, the deployed sensor node can be a malicious node directly deployed by an attacker. In order to prevent malicious nodes from joining the existing sensor network, the access control must be deployed to control sensor node deployment. By access control mechanism a deployed node can prove its identity to its neighbor nodes and also to prove that it has the right to access the existing sensor network. In order to allow authorized access of the real-time data from the sensor nodes inside WSN to the authorized users on demand, we require user access control before allowing them to access the real-time data inside WSN for which they are permitted. However, most proposed schemes are either vulnerable to different known attacks or they require high communication and computational overheads. Hence, we feel that there is a strong need to design the ideal user authentication, access control and user access control schemes, which should meet all the security requirements and achieve all the functionality requirements described in Sections 1.8, 1.9 and 1.10. These motivate us to work on these active research areas.

1.12 Objective of the work

WSN becomes a most challenging and emerging technology for the research due to its vital scope in the field coupled with their low processing power and associated low energy. User authentication, access control and user access control are the fundamental issues in designing dependable and secure WSN systems. Many different schemes are proposed for user authentication and access control for wireless sensor networks. These schemes include single password based and dynamic password based authentication, two factor authentication and biometric based authentication for user authentication in wireless sensor networks. For access control in wireless
sensor networks, different protocols have been proposed which are based on elliptic curve cryptographic techniques and the hash functions. In user access control mechanisms, different schemes are proposed in the literature, which are based on the identity-based signature, group id and user access privilege mask, and access control list. In this thesis, we aim to design novel and efficient schemes for user authentication, access control and user access control in order to provide better security for different applications in WSN.

1.13 Summary of contributions

The contributions of the thesis are summarized in the following subsections.

1.13.1 Dynamic password-based user authentication in hierarchical wireless sensor networks

The first contribution is to design a new user authentication scheme for large-scale hierarchical wireless sensor networks based on traditional passwords of users to provide user access to the real-time data by authorizing him/her directly at node level and also making it possible for users to communicate with the nodes in order to have responses to their queries. Our proposed scheme achieves better security and efficiency as compared to those for other existing password-based approaches. In addition, our scheme has merit to change dynamically the user’s password locally without the help of the base station or gateway node. Furthermore, our scheme supports dynamic nodes addition after the initial deployment of nodes in the existing sensor network. We also simulate this proposed scheme for formal security verification using the widely-accepted Automated Validation of Internet Security Protocols and Applications (AVISPA) tool. AVISPA tool ensures that whether a protocol is insecure against possible passive and active attacks, including the replay and man-in-the-middle attacks. Using the AVISPA model checkers, we show that our proposed scheme is secure against possible passive and active attacks.
1.13.2 Certificate-based access control in distributed wireless sensor networks

In our second contribution, we propose a new certificate based access control scheme for large-scale distributed wireless sensor networks, which not only identifies the identity of each node but it has also ability to differentiate between old nodes and new nodes. The proposed scheme does not require involvement of the base station during authentication and key establishment processes, and it can be easily implemented as a dynamic access control protocol. In addition, our scheme significantly reduces communication costs in order to authenticate neighbor nodes among each other and establish symmetric keys between neighbor nodes as compared with existing approaches. Further, our scheme is secure against different attacks and unconditionally secure against node capture attacks. The simulation results of our scheme using the AVISPA tool ensure that our scheme is secure.

1.13.3 Certificate-less access control in distributed wireless sensor networks

The third contribution involves to design a novel certificate-less access control scheme for large-scale distributed wireless sensor networks, which is based on elliptic curve cryptosystem. The proposed scheme is significantly better in terms of performance and security as compared to other related access control schemes. In fact, the proposed scheme requires significantly less communication costs as compared to other related schemes. Moreover, we simulate the proposed scheme for formal security analysis using the AVISPA (Automated Validation of Internet Security Protocols and Applications) tool and show that our scheme is secure.

1.13.4 User access control in hierarchical wireless body area sensor networks

The final contribution is devoted to the user access control in wireless body area networks (WBANs) used for healthcare and patient monitoring applications. We propose a novel user access control scheme for applications in wireless body area networks (WBANs). WBANs are envisioned to provide healthcare and patient monitoring applications. Without considering security, patient privacy is vulnerable in
1.14 Organization of the thesis

WBAN. Access rights for the correct information and resources for different services to the genuine users can be provided with the help of efficient user access control mechanisms. The proposed scheme makes use of the group-based user access id, access privilege mask as well as password. The elliptic curve cryptography-based public key cryptosystem is used to ensure that any particular legitimate user can access only those information for which he/she is permitted to access them. We show that the proposed scheme performs better than the previously existing user access control schemes. Through the security analysis, we show that the proposed scheme is secure against possible known attacks. Furthermore, through the formal security verification using the AVISPA tool we show that our scheme is also secure.

1.14 Organization of the thesis

The organization of the thesis is as follows.

In Chapter 1, we give an overview of wireless sensor networks. We then address the motivation and objective of our research work on user authentication, access control and user access control in wireless sensor networks. We also summarize the contributions of our research work.

In Chapter 2, we discuss the mathematical preliminaries used in our research work. We discuss in brief the one-way hash function. We then discuss the existing RSA cryptosystem including the RSA signature algorithm. We discuss the elliptic curve and its properties, the rules for point addition and scalar point multiplication over an elliptic curve, the elliptic curve digital signature algorithm, the elliptic curve discrete logarithm problem. We also give the comparison between RSA and ECC cryptosystem.

Chapter 3 gives the background information on the existing user authentication, access control and user access control in distributed as well as hierarchical wireless sensor networks, which are useful for performance comparison with our proposed schemes in these areas.

Chapter 4 presents a novel password-based dynamic user authentication scheme in hierarchical wireless sensor networks. In this chapter, we show that our scheme achieves better security and efficiency as compared to those for other existing password based approaches in WSNs.

Chapter 5 introduces a novel certificate-based dynamic access control scheme
for distributed wireless sensor networks. The proposed method is based on elliptic curve cryptosystem, hash function and symmetric-key cryptosystem, and provide better security and performances as compared to those for the existing access control schemes in WSNs.

In Chapter 6, we propose an efficient and secure certificate-less access control scheme in distributed wireless sensor networks. Our scheme provides better network performances and security as compared to the other existing access control schemes.

Chapter 7 presents a new user access control scheme in hierarchical wireless body area networks (WBANs). The proposed scheme uses a group-based user access id and access privilege mask in order to assign unique access privilege to a legitimate user for accessing the relevant information from WBANs.

Finally, in Chapter 8 we summarize the work done, highlight the contribution and suggest directions for possible future work.
Chapter 2

Mathematical Background

In this chapter, we discuss some mathematical preliminaries in order to design and analyze our proposed scheme in next sections. We discuss the properties of one-way hash function. We then describe in brief the RSA cryptosystem and RSA signature algorithm. We discuss the elliptic curve and its properties, the rules for adding points on an elliptic curve, the rules for scalar point multiplication in an elliptic curve and the elliptic curve digital signature algorithm. We also discuss the elliptic curve discrete logarithm problem. We finally compare RSA and ECC cryptosystems for their efficiency and security.

2.1 One-way hash function

A cryptographic hash function is an algorithm which accepts a variable length block of data as input and produces a fixed-size bit string, known as cryptographic hash value. Hash function can be applied to a large set of inputs which will produce outputs that are evenly distributed, and apparently random. Hash function provides data integrity. A change to any bit or bits in input data results, with high probability, in a change to the hash value. It is computationally infeasible to find either a data object that maps to a pre-specified hash result (the one-way property) or two data objects that map to the same hash result (the collision-free property). Because of these characteristics, the hash functions are often used to determine whether or not the data have changed.

Mathematically, a one-way hash function $h : \{0, 1\}^* \rightarrow \{0, 1\}^l$ takes an arbitrary-length input $x \in \{0, 1\}^*$, and produces a fixed-length (say, $l$-bits) output $h(x) \in \{0, 1\}^l$. 
\{0,1\}^l$, called the message digest or hash value. The hash function may be the fingerprint of a file, a message, or other data blocks, and has the following attributes [130].

- $h$ can be applied to a data block of all sizes.
- For any given input $x$, the message digest $h(x)$ is easy to operate, enabling easy implementation in software and hardware.
- The output length of the message digest $h(x)$ is fixed.
- Deriving the input $x$ from the given hash value $y = h(x)$ and the given hash function $h(\cdot)$ is computationally infeasible. This property is called the one-way property.
- For any given input $x$, finding any other input $y \neq x$ so that $h(y) = h(x)$ is computationally infeasible. This property is referred to as weak-collision resistant property.
- Finding a pair of inputs $(x, y)$, with $x \neq y$, so that $h(x) = h(y)$ is computationally infeasible. This property is referred to as strong-collision resistant property.

The formal definition of a one-way hash function $h(\cdot)$ is given as follows:

**Definition 2.1** (One-way hash function). As defined in [123], [132], a one-way collision-resistant hash function $h : \{0,1\}^* \to \{0,1\}^l$ is a deterministic algorithm that takes an input as an arbitrary length binary string $x \in \{0,1\}^*$ and outputs a binary string $h(x) \in \{0,1\}^l$ of fixed-length $l$. The formalization of an adversary $A$’s advantage in finding collision is as follows.

$$Adv_A^{\text{HASH}}(t) = Pr[(x,x') \leftarrow_R A : x \neq x' \text{ and } h(x) = h(x')] ,$$

where $Pr[X]$ denotes the probability of an event $X$, and $(x,x') \leftarrow_R A$ denotes the pair $(x,x')$ is selected randomly by $A$. In this case, the adversary $A$ is allowed to be probabilistic and the probability in the advantage is computed over the random choices made by the adversary $A$ with the execution time $t$. The hash function $h(\cdot)$ is said to be collision-resistant if $Adv_A^{\text{HASH}}(t) \leq \epsilon$, for any sufficiently small $\epsilon > 0$. 


There are many applications of the hash functions, for examples, in the field of cryptology and information security, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. Thus, a hash function becomes the basis of many cryptographic protocols. One fundamental property of a hash function is that its outputs are very sensitive to small perturbations in its inputs. For example, SHA-1 is a secure hash algorithm [7]. Quark [12] is a family of cryptographic hash functions proposed recently, which is designed for extremely resource-constrained environments like sensor networks and radio-frequency identification (RFID) tags. Quark can be used as a pseudo-random function (PRF), a message authentication code (MAC), a pseudo-random number generator (PRNG), a key derivation function, etc. as other hash functions are also used for these purposes. Thus, the lightweight hash function, Quark is more computationally efficient as compared to SHA-1.

2.2 RSA cryptosystem

RSA [122] is one of the popular public-key cryptosystems invented by Rivest, Shamir and Adleman in 1978 at MIT, USA. The security of RSA cryptosystem is based on the hardness of the factorization of a large positive composite integer into its prime factors.

2.2.1 RSA encryption/decryption

The RSA algorithm involves three phases, namely the key generation, encryption and decryption.

Key generation

For encrypting and decrypting a message, private and public key pair need to be generated. Each user generates a pair of public and private keys using the following steps:

Step 1. Select two large primes \( p \) and \( q \) such that \( p \neq q \). These primes \( p \) and \( q \) can be selected by the probabilistic Miller-Robin primality test algorithm [130] or the deterministic AKS algorithm [8].
Step 2. Compute the composite number \( n = p \times q \) and \( \phi(n) = (p - 1) \times (q - 1) \).

\( \phi(\cdot) \) is the Euler’s totient or phi function defined by \( \phi(n) = |\{a|0 < a < n, \gcd(a,n) = 1\}| \) as the number of positive integers less than \( n \) and relatively primes to \( n \). Note that if \( n \) is prime, \( \phi(n) = n - 1 \). For two integers \( x \) and \( y \), \( \gcd(x,y) \) can be computed using the Euclid’s algorithm [130].

Step 3. Select \( e \), with \( \gcd(e,\phi(n)) = 1 \) such that \( 1 < e < \phi(n) \).

Step 4. Compute \( d \) such that \( d \equiv e^{-1} \pmod{\phi(n)} \), i.e., \( ed \equiv 1 \pmod{\phi(n)} \). The modular inverse \( d \) can be computed using the Euclid’s extended gcd algorithm [130].

The private key consists of \( \{d,n\} \) and the public key consists of \( \{e,n\} \). The user keeps its own private key and declares the public key.

**Encryption**

Let \( A \) and \( B \) be two users, and \( A \) want to send a message \( M \) securely to another user \( B \). User \( A \) first converts \( M \) into \( k \) number of plaintext blocks \( M_i \) (\( 1 \leq i \leq k \)), such that each block \( M_i \) is an integer less than \( n \), that is, \( 0 \leq M_i < n \), using some encoding method. User \( A \) then computes the corresponding ciphertext blocks \( C_i \) (\( 1 \leq i \leq k \)), where \( C_i = M_i^e \pmod{n} \) using the public key \( (e,n) \) of the user \( B \) and the repeated square-and-multiply algorithm [130]. User \( A \) finally sends the encrypted message blocks \( C_i \) to user \( B \) via a public channel.

**Decryption**

User \( B \) can decrypt the ciphertext message \( C = [C_1, C_2, \ldots, C_k] \) by using his/her private key \( (d,n) \) to recover the original corresponding plaintext blocks \( M_i \) as \( M_i = C_i^d \pmod{n} \) using the repeated square-and-multiply algorithm [130]. After that user \( B \) concatenates all the decrypted ciphertext blocks \( M_i \) and uses the reverse encoding (decoding) method to recover the original plaintext message \( M \).

**Example 2.1:** Consider a RSA cryptosystem with \( p = 31 \), \( q = 53 \), and \( n = p \times q = 1643 \). Then \( \phi(n) = (p - 1)(q - 1) \), that is, \( \phi(1643) = (31 - 1) \times (53 - 1) = 1560 \). We select randomly \( e \), with \( \gcd(e,\phi(n)) = 1 \) such that \( 1 < e < \phi(n) \). Let \( e = 223 \). Note that \( \gcd(e,\phi(n)) = \gcd(223,1560) = 1 \). Then applying the extended Euclid’s gcd
2.2 RSA cryptosystem

Algorithm [130], we can find \( d \equiv e^{-1} \pmod{n} = 7 \). It can be easily verified that \( ed \equiv 1 \pmod{\phi(n)} \). Thus, the public key is \((e, n) = (223, 1643)\) and the private key is \((d, n) = (7, 1643)\). The encryption process begins with the conversion of the message to be sent into an integer \( M \) by means of a digital alphabet in which each letter, number, or punctuation mark of the plaintext can be replaced by a two-digit integer as \( A = 01, B = 02, \ldots, Z = 26, , = 27, . = 28, ? = 29, 0 = 30, 1 = 31, \ldots, 9 = 39, ! = 40 \), with 00 indicating a space between words. Note that this encoding technique is a standard one used in the implementation of the network security [17]. Suppose the ciphertext produced by the RSA algorithm with the public key \((e, n)\) is 1451 0103 1263 0560 0127 0897. Then, we have the ciphertext blocks as \( C_1 = 1451, C_2 = 0103, C_3 = 1263, C_4 = 0560, C_5 = 0127, C_6 = 0897 \). The deciphertext of each block \( C_i \) is as follows: \( M_i = C_i^d \pmod{n}, \forall n = 1, 2, \ldots, 6 \), using the repeated square-and-multiply algorithm [130]. We have \( M_1 = 1451^7 \pmod{1643} = 180 \), \( M_2 = 103^7 \pmod{1643} = 516 \), \( M_3 = 1263^7 \pmod{1643} = 122 \), \( M_4 = 560^7 \pmod{1643} = 500 \), \( M_5 = 127^7 \pmod{1643} = 141 \), and \( M_6 = 897^7 \pmod{1643} = 523 \). Concatenating all the messages \( M_i, (i = 1, 2, \ldots, 6) \), \( M = [M_1, M_2, M_3, M_4, M_5, M_6, M_7] = 18 05 16 12 25 00 14 15 23 = \text{“REPLY NOW”} \) becomes the original plaintext message.

2.2.2 RSA digital signature

In a digital signature algorithm, a signer signs a message using his/her own private key and any verifier can verify that signature using the public key of the signer [122]. The RSA signature algorithm consists of the key generation, signature generation and signature verification phases. These are described below [69], [122]. The key generation phase remains same as that in RSA encryption/decryption (Section 2.2.1).

**Signature generation**

Let \( A \) and \( B \) be two users and the verifier \( B \) wants to verify the signer \( A \)’s signature. Let \( m \) be a message to be signed by the user \( A \). Then \( A \) generates a signature on \( m \) using his/her own private key \( \{d, n\} \) as \( y = m^d \pmod{n} \). \( A \) sends the signed message \((m, y)\) to \( B \) via a public channel.
Signature verification

Upon receiving the signed message \((m, y)\) from the user (signer) \(A\), the verifier \(B\) verifies \(A\)'s signature using the public key \((e, n)\) of \(A\) as follows. The verifier \(B\) computes \(m' = y^e \pmod{n}\) and then verifies the condition \(m' = m\). If it is valid, \(B\) accepts \(A\)'s signature; otherwise, \(B\) rejects \(A\)'s signature.

### 2.2.3 Security of RSA cryptosystem

The security of the RSA cryptosystem is based on three mathematical approaches which are based on the problem of factoring large numbers. These approaches are the following [130]:

- Factor \(n\) into its two prime factors. This enables calculation of \(\phi(n) = (p - 1) \times (q - 1)\), which in turn, enables determination of \(d \equiv e^{-1} \pmod{\phi(n)}\).

- Determine \(\phi(n)\) directly, without first determining \(p\) and \(q\). Again, this enables determination of \(d \equiv e^{-1} \pmod{\phi(n)}\).

- Determine \(d\) directly, without first determining \(\phi(n)\).

For a large value of \(n\) with large prime factors \(p\) and \(q\), the integer factorization problem is a hard or computationally infeasible problem.

### 2.3 ECC cryptosystem

In this section, we briefly discuss the properties of an elliptic curve and ECC cryptosystem.

#### 2.3.1 Elliptic curve over finite field

Let \(a\) and \(b\) \(\in\ \mathbb{Z}_p\), where \(\mathbb{Z}_p = \{0, 1, \ldots, p - 1\}\) and \(p > 3\) be a prime, such that \(4a^3 + 27b^2 \neq 0 \pmod{p}\). A non-singular elliptic curve \(y^2 = x^3 + ax + b\) over the finite field \(GF(p)\) is the set \(E_p(a, b)\) of solutions \((x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p\) to the congruence

\[
y^2 = x^3 + ax + b \pmod{p},
\]

where \(a\) and \(b\) \(\in\ \mathbb{Z}_p\) are constants such that \(4a^3 + 27b^2 \neq 0 \pmod{p}\), together with a special point \(O\) called the point at infinity or zero point.
2.3 ECC cryptosystem

The condition $4a^3 + 27b^2 \neq 0 \pmod{p}$ is the necessary and sufficient to ensure that the equation $x^3 + ax + b = 0$ has a non-singular solution [115]. Otherwise, if $4a^3 + 27b^2 = 0 \pmod{p}$, then the corresponding elliptic curve is called a singular elliptic curve. Let $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ be two points in $E_p(a,b)$. Then $P + Q = \mathcal{O}$ implies that $x_Q = x_P$ and $y_Q = -y_P$. We have $P + \mathcal{O} = \mathcal{O} + P = P$, for all $P \in E_p(a,b)$. More precisely, a well-known theorem due to Hasse asserts that the number of points on $E_p(a,b)$, which is denoted by $\#E$, satisfies the following inequality [130]:

$$p + 1 - 2\sqrt{p} \leq \#E \leq p + 1 + 2\sqrt{p}. $$

In other words, an elliptic curve $E_p(a,b)$ over $\mathbb{Z}_p$ has roughly $p$ points on it. In addition, $E_p(a,b)$ forms an abelian or commutative group under addition modulo $p$ operation.

**Point addition on elliptic curve over finite field**

Let $G$ be the base point on $E_p(a,b)$ whose order be $n$, that is, $nG = G + G + \ldots + G(n$ times) = $\mathcal{O}$. If $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ be two points on elliptic curve $y^2 = x^3 + ax + b \pmod{p}$, $R = (x_R, y_R) = P + Q$ is computed as follows ([88], [130]):

$$x_R = (\lambda^2 - x_P - x_Q)(\pmod{p}),
$$

$$y_R = (\lambda(x_P - x_R) - y_P)(\pmod{p}),
$$

where $\lambda = \begin{cases} 
\frac{y_Q - y_P}{x_Q - x_P} \pmod{p}, & \text{if } P \neq Q \\
\frac{3x_P^2 + a}{2y_P} \pmod{p}, & \text{if } P = Q.
\end{cases}$

**Scalar multiplication on elliptic curve over finite field**

In elliptic curve cryptography, multiplication is defined as repeated additions. For example, if $P \in E_p(a,b)$, then $5P$ is computed as $5P = P + P + P + P + P$.

**Example 2.2:** Consider two points $P = (11, 3)$ and $Q = (9, 7)$ in the elliptic curve $E_{23}(1, 1)$ [26]. All the points of $E_{23}(1, 1)$ are shown in Table 2.1 as well as in Figure 2.1.

Consider two points $P = (11, 3)$ and $Q = (9, 7)$ in $E_{23}(1, 1)$. In this case, $P \neq Q$. In order to compute $R = P + Q = (x_R, y_R)$, we first compute $\lambda$ as

$$\lambda = \frac{7 - 3}{9 - 11} \pmod{23} = 21.$$
Table 2.1: Points over the elliptic curve $E_{23}(1, 1)$ (Source: [26]).

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 1)</td>
<td>(6, 4)</td>
<td>(12, 19)</td>
<td>(0, 22)</td>
<td>(6, 19)</td>
<td>(13, 7)</td>
<td>(1, 7)</td>
<td>(7, 11)</td>
<td>(13, 16)</td>
</tr>
<tr>
<td>(1, 16)</td>
<td>(7, 12)</td>
<td>(17, 3)</td>
<td>(3, 10)</td>
<td>(9, 7)</td>
<td>(17, 20)</td>
<td>(3, 13)</td>
<td>(9, 16)</td>
<td>(18, 3)</td>
</tr>
<tr>
<td>(4, 0)</td>
<td>(11, 3)</td>
<td>(18, 20)</td>
<td>(5, 4)</td>
<td>(11, 20)</td>
<td>(19, 5)</td>
<td>(5, 19)</td>
<td>(12, 4)</td>
<td>(19, 18)</td>
</tr>
</tbody>
</table>

Figure 2.1: Example of elliptic curve in case of $y^2 = x^3 + x + 1 \pmod{23}$ (Source: [26]).

Thus, $x_R$ and $y_R$ are derived as

\[
x_R = (21^2 - 11 - 9) \pmod{23} = 7,
\]
\[
y_R = (21(11 - 7) - 3) \pmod{23} = 12.
\]

As a result, $P + Q = (7, 12)$. 
To calculate $2P$, we must first derive $\lambda$ as follows:

$$\lambda = \frac{3(11^2) + 1}{2 \times 3} \pmod{23} = 7.$$

Hence, $R = P + P = (x_R, y_R)$ is computed as

$$x_R = (7^2 - 11 - 11)(\pmod{23}) = 4,$$
$$y_R = (7(11 - 4) - 3)(\pmod{23}) = 0,$$

and, thus $2P = (4, 0)$.

### 2.3.2 ECC encryption/decryption

For encryption and decryption, this cryptosystem first encodes the plaintext message $m$ to be sent as an elliptic curve point $P_m \in E_p(a, b)$. This point $P_m$ will be encrypted as a ciphertext and then subsequently decrypted.

**Key generation**

In this cryptosystem, every user $B$ has the information about the elliptic curve $E_p(a, b)$ defined over a finite field $GF(p)$, and a base point $G \in E_p(a, b)$ whose order is $n$, that $nG = \mathcal{O}$. User $A$ selects a private key $n_B$ randomly in the interval $[1, n - 1]$ and computes his/her public key $P_B = n_BG$.

**Encryption**

To encrypt a plaintext message, say $P_m$, the user $A$ first chooses a random integer $k$ in the interval $[1, n - 1]$ and produces the cipher text $C_m$ consisting of the pair of points $C_1$ and $C_2$, where $C_m = (C_1, C_2)$, with $C_1 = kG$, and $C_2 = P_m + kP_B$, where $P_B$ is the public key of user $B$. Finally, $A$ sends the encrypted message $C_m$ to user $B$ via a public channel.

**Decryption**

To decrypt the ciphertext message $C_m$, the user $B$ first multiplies the first point $C_1 = kG$ with his/her private key $n_B$ and obtains $n_B(kG) = kP_B$. $B$ then recovers the plaintext message $P_m$ as $C_2 - n_BC_1 = (P_m + kP_B) - n_B(kG) = P_m + kP_B - kP_B =$
Note that the user $A$ masks the plaintext message $P_m$ by adding $kP_B$ to it, and the value of $k$ is only known to $A$, so even though $P_B$ is a public, nobody including attackers can remove the mask $kP_B$ without knowing the value of user $B$’s private key $n_B$.

**Example 2.3:** Suppose two users $A$ and $B$ rely on the elliptic curve cryptosystem. An elliptic curve cryptosystem operates on the elliptic curve $y^2 = x^3 + ax + b \pmod{p}$ with the parameters $E_{11}(1, 1)$, where $p = 11$, $a = 1$ and $b = 6$. Note that $4a^3+27b^2 \neq 0 \pmod{11}$ and thus, the elliptic curve is non-singular. Let the base point $G$ be $G = (2, 7)$. Assume that $B$’s secret key $n_B$ is $n_B = 7$. Then, $B$’s public key becomes $P_B = n_BG = 7. (2, 7) = (7, 2)$. Let the user $A$ want to send a plaintext message $P_m = (10, 9)$ securely to the user $B$. For this purpose, let $A$ choose a random value $k = 3$. $A$ then computes the ciphertext $C_m = (C_1, C_2)$ as

\[
\begin{align*}
C_1 &= kG \\
&= 3.(2, 7) \\
&= (8, 3), \\
C_2 &= P_m + kP_B \\
&= (10, 9) + 3.(7, 2) \\
&= (10, 9) + (3, 5) \\
&= (10, 2),
\end{align*}
\]

and sends $C_m$ to $B$ via a public channel. After receiving $C_m$, user $B$ decrypts it to recover the original plaintext message $P_m$ as

\[
\begin{align*}
P_m &= C_2 - n_BC_1 \\
&= (10, 2) - 7.(8, 3) \\
&= (10, 2) - (3, 5) \\
&= (10, 2) + (3, -5) \pmod{11}, \text{ since if } P = (x_P, y_P), -P = (x_P, -y_P) \\
&= (10, 2) + (3, 6) \pmod{11} \\
&= (10, 9).
\end{align*}
\]
2.3 ECC cryptosystem

2.3.3 ECC signature

The elliptic curve digital signature algorithm (ECDSA) [82], [102] is similar to the
digital signature algorithm (DSA) [131]. The ECDSA consists of the phases: key
generation, signature generation and signature verification. These phases are de-
scribed below.

Key generation

The domain parameters for ECDSA consist of a suitably chosen elliptic cure $E_p(a,b)$
defined over a finite field $GF(p)$, and a base point $G \in E_p(a,b)$ with order $n$. Each
entity $A$ does the following:

Step 1. Select a random or pseudorandom integer $k$ in the interval $[1, n - 1]$.

Step 2. Compute $Q = kG$.

Step 3. $A$’s public key is $Q$; $A$’s private key is $k$.

Signature generation

To sign a message, say $m$, an entity $A$ with domain parameters $D = (p, n, Q, G,$
$E_p(a,b), h(\cdot))$, where $h(\cdot)$ is a secure hash function and associated key pair $(k, Q)$
does the following steps:

Step 1. Select a random or pseudorandom integer $l$, with $1 \leq l \leq n - 1$.

Step 2. Compute $lG = (x_1, y_1)$ and $r = x_1 \mod n$. If $r = 0$ then go to step 1.

Step 3. Compute $e = h(m)$ and $s = l^{-1}(e + kr) \mod n$. If $s = 0$ then go to step 1.

Step 4. $A$’s signature for the message $m$ is $(r, s)$.

Signature verification

In order to verify $A$’s signature $(r, s)$ on $m$, the verifier $B$ obtains an authentic copy
of $A$’s domain parameters $D$ and associated public key $Q$. Then $B$ does the following
steps:

Step 1. Verify that $r$ and $s$ are integers in the interval $[1, n - 1]$. 
Step 2. Compute $e = h(m)$.

Step 3. Compute $w = s^{-1} \mod n$, $u_1 = ew \mod n$, $u_2 = rw \mod n$ and $X = u_1G + u_2Q$. If $X = O$, then reject the signature. Otherwise, compute $v = x_1 \mod n$, where $X = (x_1, y_1)$.

Step 4. Accept the signature if and only if $v = r$.

### 2.3.4 Security of ECC cryptosystem

The security of ECC cryptosystem depends on the difficulty of solving the elliptic curve discrete logarithm problem (ECDLP). Before defining ECDLP, we also define the discrete logarithm problem (DLP) for better understanding.

**Discrete logarithm problem [53]**

Given an element $g$ in a finite group $S$ whose order is $n$, that is, $n = \#S_g$ ($S_g$ is the subgroup of $S$ generated by $g$) and another element $y$ in $S_g$. The problem is to find the smallest non-negative integer $x$ such that $g^x = y$. This problem, known as the discrete logarithm problem (DLP), is relatively easy to calculate discrete exponentiation $g^x \pmod{n}$ given $g$, $x$ and $n$, but it is computationally infeasible to determine $x$ given $y$, $g$ and $n$, when $n$ is large. This problem is formally defined as follows.

**Definition 2.2** (Formal definition of discrete logarithm problem [15, 42]). *Let $G$ be a cyclic group of order $n$, $g$ a generator of $G$, and $A$ an algorithm that returns an integer in $Z_n$, where $Z_n = \{0, 1, \ldots, n-1\}$. Let $a \in_R S$ denote that $a$ is chosen randomly from the set $S$. Consider the following experiment, $\text{EXP}_{G,g}^{\text{DLP}}(A)$ in Algorithm 1. The DLP-advantage of $A$ is defined by $\text{Adv}_{G,g}^{\text{DLP}}(A) = \text{Pr}[\text{EXP}_{G,g}^{\text{DLP}}(A) = 1]$, where $\text{Pr}[E]$ denotes the probability of an event $E$. The discrete logarithm problem (DLP) is said to be a hard problem in $G$ if the DLP-advantage of any adversary of reasonable resources is small, where resources are measured in terms of the time complexity of the adversary including its code size as usual. In other words, DLP is called a hard problem, if $\text{Adv}_{G,g}^{\text{DLP}}(A) \leq \epsilon$, for any sufficiently small $\epsilon > 0.$*
2.3 ECC cryptosystem

Algorithm 1 $\text{EXP}_{G,g}^{DLP}(A)$

1: Select $x \in R \mathbb{Z}_n$.
2: Compute $X \leftarrow g^x \pmod{n}$.
3: Compute $x' \leftarrow A(X)$.
4: if $g^{x'} \equiv X \pmod{n}$ then
5: return 1 (Success)
6: else
7: return 0 (Failure)
8: end if

Elliptic curve discrete logarithm problem [53]

Let $E_p(a,b)$ be an elliptic curve modulo a prime $p$. Given two points $P \in E_p(a,b)$ and $Q = kP \in E_p(a,b)$, for some positive integer $k$, where $Q = kP$ represent the point $P$ on elliptic curve $E_p(a,b)$ be added to itself $k$ times. Then the elliptic curve discrete logarithm problem (ECDLP) is to determine $k$ given $P$ and $Q$. It is computationally easy to calculate $Q$ given $k$ and $P$, but it is computationally infeasible to determine $k$ given $Q$ and $P$, when the prime $p$ is large. This problem is also formally defined as follows.

Definition 2.3 (Elliptic curve discrete logarithm problem (ECDLP) [53]). We define the elliptic curve discrete logarithm problem (ECDLP) formally as given in [53]. Let $E_p(a,b)$ be an elliptic curve modulo a prime $p$. Let $P \in E_p(a,b)$ and $Q = kP \in E_p(a,b)$ be two points, where $k \in R \mathbb{Z}_p$ (We use the notation $a \in R B$ to denote that $a$ is chosen randomly from the set $B$).

Instance: $(P, Q, r)$ for some $k, r \in R \mathbb{Z}_p$.

Output: Yes, if $Q = rP$, i.e., $k = r$, and No, otherwise.

Consider the following two distributions:

$$
\Delta_{\text{real}} = \{k \in R \mathbb{Z}_p, A = P, B = Q(= kP), C = k : (A, B, C)\},
$$

$$
\Delta_{\text{rand}} = \{k, r \in R \mathbb{Z}_p, A = P, B = Q(= kP), C = r : (A, B, C)\}.
$$

The advantage of any probabilistic, polynomial-time, 0/1-valued (false/true-valued) distinguisher $D$ in solving ECDLP on $E_p(a,b)$ is defined as

$$
\text{Adv}_{D,E_p(a,b)}^{\text{ECDLP}} = |Pr[(A, B, C) \leftarrow \Delta_{\text{real}} : D(A, B, C) = 1] - Pr[(A, B, C) \leftarrow \Delta_{\text{rand}} : D(A, B, C) = 1]|,
$$
where the probability $\Pr[.]$ is taken over the random choices of $k$ and $r$. $\mathcal{D}$ is said to be a $(t, \epsilon)$-ECDLP distinguisher for $E_p(a, b)$ if $\mathcal{D}$ runs at most in time $t$ such that $\text{Adv}^{\text{ECDLP}}_{\mathcal{D}, E_p(a, b)}(t) \geq \epsilon$.

**ECDLP assumption:** There exists no $(t, \epsilon)$-ECDLP distinguisher for $E_p(a, b)$. In other words, for every probabilistic, polynomial-time $0/1$-valued distinguisher $\mathcal{D}$, we have $\text{Adv}^{\text{ECDLP}}_{\mathcal{D}, E_p(a, b)}(t) \leq \epsilon$, for any sufficiently small $\epsilon > 0$.

### 2.4 RSA versus ECC

Compared to RSA, ECC can achieve the same level of security with smaller key size. For example, 160-bit ECC provides comparable security to 1024-bit RSA and 224-bit ECC provides comparable security of 2048-bit RSA [135]. It was pointed out in [18] that in wireless sensor networks, the transmission energy consumption rate approximately over three orders of magnitude greater than the energy consumption rates for computing. However, currently there exist few transceivers with lower communication for transmission and receiver energy consumption. An example of such transceiver is CC2420 [5]. The packet size and the number of packets in transmission will play a crucial role in the performance while designing an access control protocol in sensor networks.

Gura et al. in [71] implemented the assembly language for ECC and RSA on the Atmel ATmega 128 processor [2] and they showed in their implementation that a 160-bit point multiplication of ECC requires 0.81 s, whereas 1024-bit RSA public key operation and private key operation require 0.43 s and 10.99 s, respectively. As a result, ECC has advantages over RSA. Due to resource limitations, recent research ([77], [78], [138]) demonstrates that ECC is also much viable option in WSN as compared to RSA.

### 2.5 Summary

In this chapter, we have reviewed the basic principles of cryptography. In particular, we have discussed those cryptographic techniques which are useful for wireless sensor networks such as one-way hash function and elliptic curve cryptography. We have seen that ECC is a better approach than RSA for providing security in wireless sensor network. In addition, due to computational efficiency, the one-way hash
function and symmetric-key cryptosystem are much suited in WSN applications.
Chapter 3

Review of Related Works

In this chapter, we first provide a taxonomy of various security protocols proposed in WSN. We then give an overview of the related works on user authentication, access control and user access control in WSN.

3.1 Taxonomy of security protocols in WSNs

The key management, user authentication, access control and user authentication are the main security issues in wireless sensor networks. Figure 3.1 shows a taxonomy of security protocols in WSNs.

According to the probability of key sharing between a pair of sensor nodes, the key management schemes in WSNs can be divided into probabilistic and deterministic schemes. Pietro et al. [121] proposed a deterministic key management protocol based on the Logical Key Hierarchy (LKH). In this scheme, the base station is treated as a KDC, and all keys are logically distributed in a tree rooted at the base station. Zhu et al. [153] proposed a deterministic key management protocol, called the Localized Encryption and Authentication Protocol (LEAP), for sensor networks. Lai et al. [91] proposed a deterministic scheme, which establishes the pairwise session keys between any two neighboring nodes. It is scalable and also energy efficient. Eschenauer and Gligor [66] introduced a random key predistribution scheme for sensor networks, which relies on probabilistic key sharing among the nodes of a random graph. Chan et al. [19] proposed a \( q \)-composite scheme, where at least \( q \) common keys are needed to be shared between the key rings of nodes during the key setup phase in order to build a secure link between any two
neighboring nodes. Liu and Ning [104] proposed a polynomial pool-based key distribution scheme. The advantage of this scheme is that any two neighbor nodes can establish a secret key using the same symmetric bivariate polynomial \( f(x, y) \) of degree \( t \), and there is no extra communication overhead during the pairwise key establishment process. The main drawback is that if more than \( t \) nodes in the network are compromised by an adversary, he/she can easily reconstruct the original polynomial using Lagrange interpolation [74]. As a result, all the pairwise keys shared between the non-compromised nodes will also be compromised. Thus, this scheme is unconditionally secure and \( t \)-collusion resistant. Although increasing the value of \( t \) can improve the security property of this scheme, it is not feasible for wireless sensor networks due to the limited memory in sensors. Du et al. [64] presented another pairwise probabilistic key predistribution scheme, which is similar to Liu-Ning’s scheme [104]. However, [36], [38], [40], [50], [54] provide better security and
network performances as compared to other existing schemes.

The existing user authentications protocols proposed in the literature for wireless sensor networks usually fall into two categories: (i) password-based user authentication schemes and (ii) biometric-based user authentication schemes. According to the authentication type and factor, the protocols can be further divided into two categories: single-factor based and two-factor based. The user authentication schemes proposed by Watro et al. [141], Wong et al. [144], Tseng et al. [133], Tsern et al. [95], Ko [87] and Liu et al. [107] come under single-factor password based authentication. The user authentication schemes proposed by M. L. Das [58], Nyang et al. [116], Huang et al. [79] and Khan et al. [84] fall under two-factor password based authentication. Finally, the schemes proposed by Yuan et al. [149], and Das and Bruhadeshwar [46] fall under biometric-based user authentication. Yuan et al.’s scheme [149] uses very similar concept as in M. L. Das’s scheme and it cannot resist denial-of-service attack and node compromise attack. However, it supports freely changing password locally without contacting the GW-node in the network as compared to other password-based user authentication schemes.

Depending on the authentication type, the access control protocols are divided into two broad categories: (i) certificate-based access control schemes and (ii) certificate less access control schemes. Huang-Liu [80] proposed a certificate-less access control scheme based on the one-way hash function. The certificate-based access control schemes can be further subdivided into hash-chain based and hash-chain less access control. Huang [77] and Kim-Lee [85] proposed ECC-based access control schemes using the hash-chain. Zhou et al. [151] and Huang [78] proposed the certificate-based hash-chain less access control schemes.

To provide the access right to the genuine users for different services in terms of information and resources of sensor network, user access control is also very much essential. Wang et al. [138], Mahmud-Morogan [109] and Le et al. [93] proposed user access control based on ECC.

### 3.2 Existing user authentication schemes in WSNs

In this section, we discuss in brief the existing related user authentication schemes applicable in resource-constrained WSN environment, and their advantages and disadvantages.
Watro et al. [141] proposed a user authentication scheme for WSNs based on public-key cryptography, called the TinyPK. TinyPK uses RSA cryptosystem [122] and Diffie-Hellman scheme [60]. However, TinyPK is vulnerable to the following attack, which is described in [58], [149]. On receiving the user’s public key, an attacker can easily encrypt a session key along with other parameters and send the encrypted message to the user. After receiving the encrypted message, the user believes that the message has come from the sensor node. The user thus decrypts the receiving encrypted message using his/her private key and also uses the session key for subsequent operations the attacker intends to perform.

Wong et al. [144] proposed a user authentication scheme based on user’s password, which uses only the efficient hash function. However, there are security flaws in their scheme as described in [58], [149]. One security flaw is that their scheme does not resist many logged in users with the same login-id threat in which if an attacker possesses a valid user’s password, he/she can easily login to the sensor network. Moreover, their protocol suffers from stolen-verifier attack, because both the GW-node (base station) and login-node maintain the lookup table of the registered user’s secret information.

Tseng et al. [133] presented a modified version of Wong et al.’s scheme. It is a light-weight dynamic user authentication scheme, which allows legitimate users to choose and change their password freely, and also it does not incur extra computation cost during the password change. However, their scheme does not provide mutual authentication between the GW-node and a sensor node. Moreover, this scheme cannot resist the wormhole and node replication attacks.

Tsern et al. [95] proposed a strong-password based dynamic user authentication protocol. However, their scheme also suffers from mutual authentication problem between the GW-node and a sensor node. As a result, this scheme does not resist the wormhole, Sybil and node replication attacks.

M. L. Das proposed an efficient scheme [58] based on passwords, which improves the security over Wong et al.’s scheme. This scheme uses timestamp for authentication. However, it cannot resist denial-of-service attack and node compromise attack. Another drawback of this scheme is that a user cannot securely and freely change his/her password locally without contacting the GW-node. Later, Khan and Alghathbar [84] showed that M. L. Das’s scheme [58] is insecure. Khan and Alghathbar showed that M. L. Das’s scheme is insecure against GW-node bypassing attack.
Further, Khan and Alghathbar showed that M. L. Das’s scheme is insecure against privileged-insider attack due to the following reason. In M. L. Das’s scheme, the user $U_i$ performs registration phase with the GW-node by presenting his/her password $PW_i$ in plaintext format. Hence, if the system manager or a privileged-insider of the GW-node knows the password of $U_i$, he/she may try to impersonate $U_i$ by accessing other servers, where $U_i$ could be also a registered user because the user $U_i$ may use the same password to access different applications or servers for his/her convenience of remembering long password and easy-to-use whenever required. Some more improvements and enhancements of M. L. Das’s scheme have been proposed in [79], [116].

He et al. [73] proposed an enhanced scheme based on M. L. Das’s scheme [58]. Their scheme keeps the original merits of M. L. Das’s scheme and can withstand the security weaknesses such as vulnerabilities to insider and impersonation attacks.

Vaidya et al. [134] showed that M. L. Das’s scheme [58] and Khan-Alghathbar’s scheme [84] have security flaws, and remain vulnerable to various attacks including stolen smart card attacks. In order to overcome security weaknesses of both schemes [58] and [84], they proposed an improved two-factor user authentication, which is resilient to stolen smart card attacks as well as other common type of attacks.

Fan et al. [68] proposed a simple user authentication scheme, which is efficient and Denial-of-Service (DoS) resistant user authentication scheme for two-tiered WSNs. Their scheme can establish a session key between the user and a master node (cluster node) in the sensor network.

Chen and Shih [23] pointed out that M. L. Das’s scheme [58] fails to achieve mutual authentication. To tackle such problem, they proposed a robust mutual authentication protocol for WSNs.

Recently, biometric-based user authentication in WSNs has also drawn some research attention. A biometric-based user authentication scheme for WSNs has been proposed by Yuan et al. in [149]. It uses very similar concept as in M. L. Das’s scheme. Yuan et al.’s scheme does not also offer any protection against denial-of-service attack, because the GW-node does not expect any acknowledgment from a sensor node. Moreover, the GW-node and the sensor nodes will not know about a message, if an attacker blocks it from reaching the nodes. Further, their scheme is not resilient against node compromise attack. However, it supports freely changing password locally by the user without contacting the GW-node in the network.
46 Review of Related Works

as compared to other schemes [58], [141], [144]. In 2013, Das and Bruhadeshwar [46] have proposed a new biometric-based user authentication mechanism in heterogeneous (hierarchical) wireless sensor networks. Their scheme provides strong authentication compared to traditional related password-based schemes and achieves good properties such as it works without synchronized clock, it freely changes the user’s password, it provides low computation costs and mutual authentication. Their scheme establishes a symmetric secret session key shared between the user and a sensor node so that the secret session key can be used later for secure future communications between them. Moreover, their scheme provides unconditional security against node capture attack and it is also resilient against different attacks. Overall, their scheme has better network performances as compared to other existing password-based schemes and Yuan et al.’s biometric-based scheme [149].

3.3 Existing access control schemes in WSNs

In this section, we discuss the following existing related access control schemes proposed in sensor networks.

Zhou et al. [9] proposed an access control scheme, which is based on elliptic curve cryptographic techniques for sensor network. Their scheme is more efficient than the schemes based on RSA. Their scheme consists of the following phases. In pre-deployment phase, before a sensor network is deployed, the certificate authority (CA) chooses a set of network parameters and preloads a set of node parameters to each sensor node. In node deployment phase, sensor nodes bootstrap themselves and then start establishing communications among them. During the network operation, if some nodes are lost due to power exhaustion problem or some nodes are detected as malicious, then new nodes need to be deployed in the target field. Each new node has a preset bootstrapping time different from that of the previously deployed nodes. In node authentication phase, every new node broadcasts a message to inform its neighbors for its existence. In this phase, there are two kind of handshakes between nodes: the handshake between new nodes, and the handshake between a new node and an old node. The purpose of these handshakes is to authenticate each node with its neighbor nodes as well as to establish secret keys between neighbor nodes. Zhou et al.’s scheme thus supports new nodes joining in the sensor network dynamically and it supports the key establishment in peer to peer manner by using the bootstrapping
time in the preloaded certificates present in the nodes. However, the drawback of
their scheme is that it introduces high communication overheads due to exchange
of so many messages for the entire protocol during node authentication and key
establishment phase.

Huang [122] proposed an efficient access control protocol based on elliptic curve
cryptography (ECC) and hash-chain. This scheme is suitable for resource-constrained
sensor nodes and could be easily implemented as a dynamic access control because
all the old secrets and broadcasting information in existing deployed nodes should
not be updated once a new node is added. This scheme requires involvement of the
base station during the initialization phase, and also node authentication and key
establishment phase. The limitation of this scheme is that it may not support a
large-scale network and thus, it may not be scalable.

Kim and Lee [60] showed that Huang’s scheme [122] is insecure against the replay
attack and an active attack known as new node masquerading attack, and has the
lack of hash chain renewability. In order to remedy the weaknesses in Huang’s
scheme, Kim and Lee [60] proposed an enhanced access control protocol over sensor
networks. Their scheme consists of the initialization phase, the authentication and
key establishment phase, and the new node injection phase. They have used the
renewal of hash chain phase to overcome weaknesses in Huang’s scheme for the
hash chain exhausted nodes. Due to renewal of hash chain, existing nodes need to
communicate with the base station and as a result, it introduces high communication
overheads. Similar to Huang’s scheme, Kim-Lee’s scheme is also not scalable to
support a large-scale sensor network. However, Shen et al. [127] showed that their
scheme is also vulnerable to a fatal weakness, where their scheme is insecure against
an active attack, called the man-in-the-middle attack.

Huang [78] proposed a simple dynamic access control protocol to prevent malici-
ous nodes from joining sensor networks. This scheme [78] uses the existing Schnorr
signature [124] during the authentication phase. This scheme also uses the expira-
tion time for each deployed sensor node so that once the time period elapses, the
sensor nodes in the network cannot access any data for future time period. How-
ever, if an adversary captures a sensor node and deploys another fake node using
the captured node’s information, the deployed fake node can still authenticate and
establish successfully with its neighbor nodes until expiration time period elapses.
Further, this scheme requires high storage and computational overheads.
3.4 Existing user access control schemes in WSNs

In this section, we discuss briefly the existing related user access control schemes, which are proposed in resource-constrained wireless sensor networks.

Wang et al. [139] split the access control process into local authentication conducted by a group of sensors physically close to a user, and proposed a remote authentication based on the endorsement of the local sensors. They implemented the access control protocol on a testbed of TelosB motes [2]. Based on ECC, they provided the local authentication. In this scheme, using certificate-based authentication, the user access is verified by the sensor nodes.

He et al. [72] proposed a distributed privacy-preserving access control scheme for WSNs. They identified the characteristics of a single-owner multi-user sensor network and the requirements of distributed privacy-preserving access control. Their scheme is based on ring signature technique. The user first registers to the network owner. The network owner then divides all users into groups. The same group has the same access privilege. The network owner maintains a group access list pool, which contains the identity and other information of each group and based on the group, the access control is provided.

Wen et al. [142] proposed a user access control scheme for wireless multimedia sensor network. In this scheme, the authorized user can access the real-time multimedia data. Their proposed scheme uses on the Chinese Remainder Theorem-based group rekeying mechanism.

Li et al. [99] discussed various practical issues required to fulfill the security and privacy requirements in wireless body area sensor networks (WBANs). They explored the relevant security solutions in sensor networks and WBANs, and also analyzed various applications. They proposed an attribute-based encryption for achieving fine-grained access control. This is a one-to-many encryption method, where the ciphertext is readable only by a group of users that satisfy a certain access policy.

Mahmud and Morogan [109] proposed an identity-based user authentication and access control protocol based on the identity-based signature (IBS) scheme. They used ECC-based digital signature algorithm (DSA) for signing and verifying a message. At the time of initialization, sensor nodes and users registered to the base station, and group identity and access rights of the users are also given by the base
station. User revocation is done by expiration of access time of the user assigned by the base station at the time of registration. The authenticated user is not allowed to get the requested access without having the proper access right. Though their scheme is secure against node capture and denial-of-service (DoS) attack, but password change process is not supported. For a new user addition, the base station needs to broadcast again the user’s parameters such as user id, group id and system timestamp, which incur more communication overhead in the network.

Wang et al. [138] proposed an ECC-based user access control scheme. In this scheme, before authentication the user needs to register to the key distribution center (KDC) for the access permission. KDC maintains a user access list pool with the respective user’s access privilege. This access privilege consists of user id, group id and user access privilege mask. The multiple users within a same group should have the same access privilege. Based on elliptic curve cryptography, in this scheme the KDC generates the public key, the private key of the user and the certificate of user access list based on the user’s request. The user requests the sensor node by sending its certificate and after that the sensor node selects one random number as a session key. In this scheme, the user authenticates a sensor node and a sensor node also authenticates the user so that the mutual authentication is provided between the user and the sensor node.

Le at al. [93] proposed an energy-efficient access control scheme based on ECC. In this scheme, they proposed an improvement over Wang et al’s scheme [138]. This scheme is a public-key cryptography based access control scheme, where the user has to take access permissions from a key distribution center (KDC). KDC maintains an access control list (ACL) pool and associated user identifications. User’s access privileges are defined in ACL based on user access privilege mask. The public keys between the KDC and the sensor nodes are mutually exchanged during the pre-deployment phase. After registration, the user gets his/her public key and private key. One signed certificate of the access control list is also issued by the KDC and sent to the user. The user then needs to be authenticated by the sensor node for future communications.
3.5 Summary

In this chapter, we have presented an overview of state of the art of the related works in the areas of sensor network security, such as user authentication, access control and user access control, which are available up to date in the literature. However, it is noted that most schemes proposed in the literature are either vulnerable to different attacks or they require high computational overheads.
As most of the applications in wireless sensor network (WSN) are real-time based, so users are generally interested in accessing real-time information. This is possible if the users (called the external parties) are allowed to access the real-time data directly from the nodes inside WSN and not from the base station BS. Usually, the information from nodes are gathered periodically in the BS and so, the gathered information may not be real-time. In order to get the real-time information from the nodes, the user needs to be first authorized to the nodes as well as the BS so that illegal access to nodes do not happen. As a result, the user authentication problem becomes a very important topic in research of WSN security. Consider an example for critical applications in medical system, which is shown in Figure 4.1. In a wireless body area network (WBAN), some sensors are deployed in a patient’s body for measuring medical data such as ECG, body movement, temperature, respiration, heart rate, pulse oximeter, blood pressure, blood sugar, etc. These sensors transmit sensing data in a secure channel to a small body area network gateway. The gateway then processes data locally and resends them through a secure channel to the router for the external network to the medical server at the hospital. The results are then observed and analyzed by the medical staffs/doctors to monitor patients. Since the patients data are confidential, only authorized users must be given access to monitor the patient. In order to access the real time data from the patient’s body,
the external users (in this example, the doctors) must be authenticated by the base station (medical server) and the sensors before allowing to the access to the sensing data from the sensors inside WBAN.

Figure 4.1: Security measures at gateway in wireless body area networks (Source: [11], [44]).

4.1 The proposed password-based user authentication scheme

In this section, we first discuss the network model and threat model used in our scheme. We then give the list of notations used in our proposed scheme. Finally, we describe the different phases related to our scheme.

4.1.1 Network model

We consider the hierarchical or heterogeneous wireless sensor network (HWSN) model, shown in Figure 4.2, for developing our proposed scheme due to the following reasons [24]. Wireless sensor networks are distributed event-driven systems that differ from traditional wireless networks in several ways, for examples, extremely large network size, severe energy constraints, redundant low-rate data, and many-to-one flows. In many sensing applications, connectivity between all sensor nodes is not
always necessary. As a result, data centric mechanisms can be performed to aggregate redundant data in order to reduce the energy consumption and traffic load in wireless sensor networks, and thus, HWSN has more operational advantages than the distributed WSN model (DWSN), shown in Figure 1.2 (Chapter 1), for wireless sensor networks because of inherent limitations of sensors on power and processing capabilities.

![Hierarchical Wireless Sensor Network (HWSN) Architecture](image)

Figure 4.2: A hierarchical wireless sensor network (HWSN) architecture.

### 4.1.2 Threat model

Due to the hostile environments in the deployment field, nodes can be physically captured by an attacker. We assume that both the sensor nodes as well as cluster heads can be compromised or captured by an attacker. Usually, nodes are not equipped with tamper-resistant hardware due to cost constraints and hence, we assume that once a node is captured by an attacker, all the stored sensitive data as well as cryptographic information are revealed to the attacker [111]. However, we assume that in any case, the base station (BS) will not be compromised by an attacker. Finally, as in [58], we make use of the famous Dolev-Yao threat model [61] in which two communicating parties (nodes) communicate over an insecure channel. We adopt the similar threat model for WSNs where the channel is insecure and the end-points (users, sensor nodes, cluster heads) cannot in general be trustworthy.

Eschenauer and Gligor proposed a centralized node revocation method [66], in which when the base station detects a misbehaving node, it broadcasts a message to revoke that node. A localized mechanism for sensor network node revocation was
further proposed by Chan, Perrig and Song [19] and in their approach, nodes can revoke their neighbors. The sybil attack in sensor network has been analyzed and described by Newsome et al. [114]. Further, a mechanism for distributed detection of node replication attacks in sensor networks was proposed by Parno et al. in [120]. Zhu et al. proposed an approach [152] that combines deterministic mapping (to reduce communication and storage costs) with randomization (to increase the level of resilience to node compromise). Their approach performs better than Parno et al.’s approach [120]. We thus assume that the compromised (captured) nodes can be detected and as a result, the base station, cluster head and sensor nodes in each cluster know the ids of the compromised nodes. Consequently, the base station alerts the users with the compromised cluster heads in the network.

4.1.3 Notations

We use the notations in this chapter to describe our proposed scheme in Table 4.1.

The hash functions have many applications in the field of cryptology and information security, notably in digital signatures, message authentication codes (MACs), and other forms of authentication and thus, it becomes the basis of many cryptographic protocols. One of the fundamental properties of hash functions is that the outputs are very sensitive to small perturbations in their inputs. We use SHA-1 as the secure hash algorithm [7]. For efficient encryption/decryption, we use the symmetric key AES algorithm [1], [27].

4.1.4 Our contributions

In this chapter, we propose a new user authentication scheme based on traditional passwords of users to provide user access to the real-time data by authorizing him/her directly at node level and also making it possible for users to communicate with the nodes in order to have responses to their queries. Our scheme has the following attractive properties:

- It provides better security as compared with the other related schemes, since it supports mutual authentication between the user and the cluster heads, resists denial-of-service attack, privileged-insider attack, smart card breach attack and node capture attack.
4.1 The proposed password-based user authentication scheme

Table 4.1: Notations used in this chapter.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_i$</td>
<td>$i^{th}$ User</td>
</tr>
<tr>
<td>$BS$</td>
<td>Base station</td>
</tr>
<tr>
<td>$S_j$</td>
<td>Sensor node</td>
</tr>
<tr>
<td>$CH_j$</td>
<td>Cluster head in the $j$-th cluster</td>
</tr>
<tr>
<td>$PW_i$</td>
<td>Password of a user $U_i$</td>
</tr>
<tr>
<td>$ID_i$</td>
<td>Identity of a user $U_i$</td>
</tr>
<tr>
<td>$ID_{CH_j}$</td>
<td>Identifier of cluster head $CH_j$</td>
</tr>
<tr>
<td>$h(\cdot)$</td>
<td>A secure one-way collision-resistant hash function</td>
</tr>
<tr>
<td>$E_k(\cdot)$</td>
<td>Symmetric key encryption algorithm using the key $k$</td>
</tr>
<tr>
<td>$D_k(\cdot)$</td>
<td>Symmetric key decryption algorithm using the key $k$</td>
</tr>
<tr>
<td>$X_s$</td>
<td>A secret information maintained by the base station</td>
</tr>
<tr>
<td>$X_A$</td>
<td>A secret information shared between a user $U_i$ and the base station</td>
</tr>
<tr>
<td>$y$</td>
<td>A secret random number known to a user $U_i$ and the base station</td>
</tr>
<tr>
<td>$T$</td>
<td>Timestamp</td>
</tr>
<tr>
<td>$\Delta T$</td>
<td>Maximum transmission delay or preset acceptable delay threshold or expected time interval for the transmission delay or expected network delay time</td>
</tr>
<tr>
<td>$A</td>
<td></td>
</tr>
<tr>
<td>$A \oplus B$</td>
<td>XOR operation of $A$ and $B$</td>
</tr>
</tbody>
</table>

- It supports dynamic node addition after initial deployment of nodes in the network. The proposed scheme does not require to update information for new nodes addition in the user’s smart card.

- It supports changing the user’s password locally without the help of the $BS$.

- It provides unconditional security against node capture attacks. That is, compromise of a cluster head does not reveal any secret information of other cluster heads and it does not lead to compromise any other secure communication between the user and the non-compromised nodes in the network.
• It establishes a secret session key between the user and a cluster head for future secret communication of the real-time data inside WSN between them using the established session key.

• We have then analyzed the security of our scheme using the formal security under the random oracle models and the formal security validation using the widely-accepted AVISPA model checker, OFMC. The formal security under the random oracle models reveals that our scheme is secure. Furthermore, we have simulated our scheme for the formal security verification using the widely-accepted AVISPA tool. Finally, we have implemented other schemes in the HLPSL language and then compared our scheme for formal security verification using AVISPA OFMC model checker with other existing schemes.

• In addition, we have compared the functionality provided by our scheme with other schemes. Overall, our proposed scheme has better performance than other existing schemes.

4.2 Description of the proposed scheme

In this section, we discuss our proposed password based dynamic user authentication scheme. Our scheme consists of seven phases: pre-deployment phase, post-deployment phase, registration phase, login phase, authentication phase, password change phase and dynamic node addition phase. Since our scheme is based on timestamp mechanism to prevent the replay attack, we assume that all the nodes (sensors, cluster heads and BS) are synchronized with their clocks.

We consider a hierarchical wireless sensor network (HWSN) \cite{36}, \cite{55} consisting of two types of sensors: a small number of powerful High-end sensors (H-sensors) and a large number of resource-constrained Low-end sensors (L-sensors). The H-sensors can execute relatively complicated numerical operations than the L-sensors. The H-sensors have much larger radio transmission range and also larger storage space than the L-sensor nodes. The L-sensors are extremely resource-constrained. For example, the H-sensors can be Personal Digital Assistants (PDAs) and the L-sensors are the MICA2-DOT motes \cite{6}. Since MICA2 motes are now obsolete devices, one can use the L-sensors as MICAz/IRIS sensor devices \cite{6}. Dong and Liu \cite{62} used the TelosB devices \cite{6}, which have 1MB flash memory, as assisting nodes in the network. The
4.2 Description of the proposed scheme

assisting nodes act as cluster heads to facilitate pairwise key establishment between
sensor nodes in the network. Further, one can also use the SunSPOT (Sun Small
Programmable Object Technology) devices [129] as cluster heads. The SunSPOT
device is built upon the IEEE 802.15.4 standard and it supports the IEEE 802.15.4
MAC layer, on top of which e.g. Zigbee [154] can be built. After deployment of
cluster heads and sensor nodes, the sensor nodes communicate among their neighbor
sensor nodes in a cluster. The sensor nodes also communicate with the neighbor
cluster head in that cluster. The cluster heads communicate among each other and
also with the base station. All these communication take place using the IEEE
802.15.4 standard.

The target field is considered as two dimensional and it is partitioned into a
number $m$ of equal sized disjoint clusters. Each cluster consists of a cluster head
$CH_j$ (here it is an H-sensor node) and a number $n_i$ of L-sensor nodes. For the sake
of simplicity, we call an L-sensor node as a regular sensor node and an H-sensor node
as a cluster head (CH). The number $n_i$ of regular sensor nodes is to be taken in each
cluster so that the network connectivity in each cluster is high so that every sensor
node can communicate securely among each other and finally with their neighbor
cluster head in that cluster. The sensors are to be deployed randomly in a cluster
and each cluster head is deployed in that cluster around the center of that cluster.
The base station (BS) can be located either in the center or at a corner of the
network.

In our proposed scheme, we make use of the timestamp to protect the replay
attack from an adversary. Thus, we assume that all the nodes (sensor nodes, cluster
heads and BS) are synchronized with the clocks during their deployment in a target
field.

4.2.1 Pre-deployment phase

The (key) setup server (the base station) performs the following steps in offline before
deployment of the sensor nodes and cluster heads in a target field (deployment field):

- **Step 1:** The setup server assigns a unique identifier, say $ID_{CH_j}$ to each cluster
  head $CH_j$ which will be deployed in the target field. For each deployed regular
  sensor node $S_i$, the setup server also assigns a unique identifier, say $ID_{S_i}$.

- **Step 2:** The setup server then selects randomly a unique master key, say
$MK_{CH_j}$ for each cluster head $CH_j$. Note that the master key $MK_{CH_j}$ is shared between the cluster head $CH_j$ and the base station only. Similarly, the setup server also assigns a unique randomly generated master key, say $MK_{S_i}$ for each deployed regular sensor node $S_i$, which will be shared with the base station only.

- **Step 3:** Finally, the setup server loads the following information into the memory of each cluster head $CH_j$ ($j = 1, 2, \ldots, m$): (i) its own identifier, $ID_{CH_j}$ and (ii) its own master key $MK_{CH_j}$. Each deployed regular sensor node $S_i$ in the cluster $C_j$ is loaded with the following information: (i) its own identifier, $ID_{S_i}$ and (ii) its own master key $MK_{S_i}$.

### 4.2.2 Post-deployment phase

As soon as regular sensor nodes are deployed randomly in their respective clusters, their task is to locate the physical neighbors within their communication ranges. For this purpose, each sensor node broadcasts a HELLO message containing its own identifiers to other nodes in its communication range. Upon receiving other HELLO messages each sensor node prepares a list of its neighbor nodes. Cluster heads in their own clusters locate their physical neighbors which are the regular sensor nodes. Cluster heads also locate their other cluster heads in their communication ranges in the network.

For secure communication between neighbor regular sensor nodes, and between neighbor regular sensor nodes and cluster head in a cluster, nodes require to establish pairwise secret keys between them. Since our main goal in this chapter is user authentication, so we assume that nodes in a cluster can establish secret keys using some existing efficient and secure key establishment techniques. Note that the purpose of pre-loading the master key $MK_{S_i}$ into a sensor node $S_i$’s memory prior to its deployment in a particular cluster $C_j$ having the cluster head $CH_j$ is that after its deployment in a target field that sensor node $S_i$ needs to establish the pairwise secret symmetric keys with its neighbors in that cluster $C_j$ for secure communications. For this, we have used the unconditionally secure key establishment scheme [36] for pairwise key establishment between neighbor nodes in each cluster and between cluster heads in the network. The key establishment scheme [36] is a group-based deterministic key pre-distribution scheme, which uses the master key
4.2 Description of the proposed scheme

$MK_S$ of a sensor node $S_i$ for establishing the secret pairwise keys with its neighbor sensor nodes as well as the cluster head $CH_j$ in the cluster $C_j$. It always guarantees that a direct secret symmetric key is established between any two neighbor nodes in a cluster. Further any two neighbor cluster heads establish a secret key between them, and as a result this scheme provides 100% secure network connectivity. For details, one can refer to [36].

This scheme also guarantees that no matter how many sensor nodes are compromised by an attacker, the non-compromised node still can communicate with 100% secrecy, which is called the unconditionally secure or perfectly security property against node capture attacks. In addition, this scheme provides significantly better trade-off between communication overhead, computational overhead, storage overhead, network connectivity and security against node capture attacks as compared to other existing key pre-distribution schemes [16], [19], [24], [66], [83], [105], [106].

After key establishment, sensor nodes can securely communicate with other neighbor sensor nodes and their cluster head in the cluster. Cluster heads can also securely communicate with other neighbor cluster heads and finally to the base station.

4.2.3 Registration phase

When the remote user authentication scheme starts, the user $U_i$ and the base station $BS$ need to perform the following steps:

- **Step 1:** The user $U_i$ selects a random number $y$, the identifier $ID_i$ and the password $PW_i$. $U_i$ then computes $RPW_i = h(y||PW_i)$. $U_i$ provides the computed masked password $RPW_i$, the identity $ID_i$ and the secret random number $y$ to the base station via a secure channel.

- **Step 2:** The $BS$ then randomly generates a secret number $X_A$ for the user $U_i$ and computes $f_i = h(ID_i||X_s)$, $x = h(RPW_i||X_A)$, $r_i = h(y||x)$, and $e_i = f_i \oplus x = h(ID_i||X_s) \oplus h(RPW_i||X_A)$. Note that the secret information $X_s$ is only known to the $BS$, whereas the secret information $X_A$ is shared between the user $U_i$ and the BS. Thus, each randomly generated secret number $X_A$ is distinct for each user $U_i$ in the network.

- **Step 3:** The $BS$ then selects all $m$ deployed cluster heads in the network,
CH₁, CH₂, ..., CHₘ, which will be deployed during the initial deployment phase, and computes the m key-plus-id combinations \{ (Kₖ, IDₖCH) \mid 1 \leq j \leq m \}, where \( K_j = E_{MₖCH_j}(ID_i||IDₖCH||X_s) \). Note that the master key \( MₖCH_j \) is already loaded in the cluster head \( CH_j \)’s memory prior to its deployment in the network in Step 2 of our pre-deployment phase (Section 4.2.1). The encryption of \( ID_i, IDₖCH \) and \( X_s \) using the master key \( MₖCH_j \) leads to distinct key \( K_j \) shared between each user \( U_i \) and \( CH_j \). This key \( K_j \) is computed on the fly by the corresponding cluster head \( CH_j \) using its own pre-loaded master key \( MₖCH_j \) during the authentication phase (Section 4.2.5).

- **Step 4:** For dynamic cluster head addition phase, assume that another \( m'(\leq m) \) cluster heads, \( CH_{m+1}, CH_{m+2}, ..., CH_{m+m'} \), be deployed later after the initial deployment in the network in order to replace some compromised cluster heads, if any, and add some fresh cluster heads along with sensor nodes. For this purpose, the BS further computes another \( m' \) key-plus-id combinations \{ (Kₘ₊ₗ, IDₘ₊ₗCH) \mid 1 \leq j \leq m' \}, where \( K_{m+j} \) is computed as \( K_{m+j} = E_{MₖCH_{m+j}}(ID_i||IDₖCH_{m+j}||X_s) \). \( IDₘₖCH_{m+j} \) is the unique identifier generated by the BS for the cluster head \( CH_{m+j} \) to be deployed during the dynamic node addition phase and \( MₖCH_{m+j} \) the unique master key randomly generated by the BS for \( CH_{m+j} \), which is shared between it and the BS.

- **Step 5:** Finally, the BS generates a tamper-proof smart card with the following parameters: (i) \( ID_i \), (ii) \( y \), (iii) \( X_A \), (iv) \( r_i \), (v) \( e_i \), (vi) \( h(\cdot) \), and (vi) \( m + m' \) key-plus-id combinations \{ (Kₖ, IDₖCH) \mid 1 \leq j \leq m + m' \}.

Note that initially \( m \) cluster heads will be deployed in the network. Therefore, an adversary can compromise at most \( m \) cluster heads in the network, and as a result, we need to re-deploy \( m' (\leq m) \) cluster heads in that network after initial deployment. The value of \( m + m' \) is chosen according to memory availability of the smart card. For example, we can store \( m + m' = 200 \) encrypted keys along with the identifies of the cluster heads in the memory of a smart card. We have only a small number of cluster heads to be deployed for a large-scale wireless sensor network along with a large number of regular sensor nodes. Thus, if each cluster contains 220 sensor nodes, then for a hierarchical sensor network containing 22,000 regular sensor nodes we only require 100 cluster nodes in the network. Thus, it is a practical assumption to store the computed \( m = 100 \) encrypted keys for initial
4.2 Description of the proposed scheme

Deployment of cluster heads and \( m' = 100 \) encrypted keys for dynamic cluster heads addition so that \( m + m' = 200 \) encrypted keys can be stored into the memory of the smart card. The base station also deletes the secret random number \( y \) for security purpose. Hence, the secret random number \( y \) will be used by the user \( U_i \) only, after the registration phase.

Further, note that the \( m + m' \) encrypted keys stored into the memory of the smart card of a user \( U_i \) are different from those for another user \( U_j \), because these keys are encrypted using the master keys of cluster heads along with the different identifiers of users, the identifiers of cluster heads and the secret information \( X_s \).

This registration phase is summarized in Table 4.2.

Table 4.2: Registration phase of our proposed scheme.

<table>
<thead>
<tr>
<th>( U_i )</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Selects ( ID_i, PW_i ), and ( y ). Computes ( RPW_i = h(y</td>
<td></td>
</tr>
<tr>
<td>2. Computes ( f_i = h(ID_i</td>
<td></td>
</tr>
</tbody>
</table>

4.2.4 Login phase

If the user \( U_i \) wants to access the real-time data from the WSN, the user \( U_i \) needs to perform the following steps:
• **Step 1:** $U_i$ first inserts his/her smart card into the card reader of a specific terminal and provides his/her password $PW_i$.

• **Step 2:** The smart card then computes the masked password of the user $U_i$ as $RPW'_i = h(y||PW_i)$. Using the computed masked password, the smart card further computes $x' = h(RPW'_i||X_A)$ and $r'_i = h(y||x')$, and then verifies whether $r'_i = r_i$. If this verification does not hold, it means that $U_i$ has entered his/her password $PW_i$ incorrectly and the scheme terminates immediately. Otherwise, the smart card performs the following steps.

• **Step 3:** Using the system’s current timestamp $T_1$, the smart card computes the hash value $N_i = h(x'||T_1)$.

• **Step 4:** The user $U_i$ selects a cluster head, say $CH_j$ from which the real-time data can be accessed inside WSN. Corresponding to $CH_j$, the smart card selects the encrypted master key of $CH_j$, $K_j$ from its memory and computes a ciphertext message $E_{K_j}(ID_i||ID_{CH_j}||N_i||e_i||T_1)$. Finally, the user $U_i$ sends the message $\langle ID_i||ID_{CH_j}||E_{K_j}(ID_i||ID_{CH_j}||N_i||e_i||T_1) \rangle$ to the $BS$, via a public channel.

The login phase is summarized in Table 4.3.

<table>
<thead>
<tr>
<th>$U_i$</th>
<th>$BS$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Inserts the smart card and inputs $PW_i$.</td>
<td></td>
</tr>
<tr>
<td>2. Computes $RPW'_i = h(y</td>
<td></td>
</tr>
<tr>
<td>3. Verifies if $r'_i = r_i$? If it holds, computes $N_i = h(x'</td>
<td></td>
</tr>
<tr>
<td>4. $\langle ID_i</td>
<td></td>
</tr>
</tbody>
</table>
4.2.5 Authentication phase

After receiving the login request message \( \langle ID_i||ID_{CH_j} || E_{K_{CH_j}}(ID_i||ID_{CH_j}||N_i || e_i||T_1) \rangle \) from the user \( U_i \), the BS performs the following steps in order to authenticate the user \( U_i \):

- **Step 1**: The BS computes a symmetric key \( K \) using the stored master key \( MK_{CH_j} \) of the cluster head \( CH_j \) as \( K = E_{MK_{CH_j}}(ID_i||ID_{CH_j}||X_s) \). Using this computed key \( K \), the BS decrypts \( E_{K_{CH_j}}(ID_i||ID_{CH_j}||N_i || e_i||T_1) \) for the information \( ID_i, ID_{CH_j}, N_i, e_i \) and \( T_1 \) as \( D_{K}(E_{K_{CH_j}}(ID_i||ID_{CH_j}||N_i || e_i||T_1)) = (ID_i||ID_{CH_j}||N_i || e_i||T_1) \).

- **Step 2**: The BS checks if the retrieved \( ID_i \) is equal to the received \( ID_i \) and also if the retrieved \( ID_{CH_j} \) is equal to the received \( ID_{CH_j} \). If these hold, the BS further checks if \( |T_1 - T_1^*| \leq \Delta T_1 \), where \( T_1^* \) is the current system timestamp of the BS or the time when the BS receives the login request message \( \langle ID_i||ID_{CH_j} || E_{K_{CH_j}}(ID_i||ID_{CH_j}||N_i || e_i||T_1) \rangle \) from the user \( U_i \), and \( \Delta T_1 \) denotes the expected time interval for the transmission delay [20], [57], [58] or the maximum transmission delay or the preset acceptable delay threshold [94] or expected network delay time [130]. Note that \( \Delta T_1 \) is the empirical value obtained from the experiments depending on one network to another network, and it is usually a small value for the security applications in WSNs. Now, if this condition holds, the BS further computes \( X = h(ID_i||X_s), Y = e_i \oplus X \), and \( Z = h(Y||T_1) \). If \( Z = N_i \), then the BS accepts \( U_i \)'s login request and \( U_i \) is considered as a valid user by the BS. Otherwise, the scheme terminates.

- **Step 3**: Using the current system timestamp \( T_2 \), the BS computes \( u = h(Y||T_2) \) and produces a ciphertext message encrypted using the master key \( MK_{CH_j} \) of the cluster head \( CH_j \) as \( E_{MK_{CH_j}}(ID_i||ID_{CH_j}||u||T_1||T_2||X||e_i) \). The BS sends the message \( \langle ID_i||ID_{CH_j} || E_{MK_{CH_j}}(ID_i || ID_{CH_j} || u||T_1||T_2||X||e_i) \rangle \) to the corresponding cluster head \( CH_j \).

- **Step 4**: After receiving the message in Step 3 from the BS, the cluster head \( CH_j \) decrypts \( E_{MK_{CH_j}}(ID_i || ID_{CH_j} || u||T_1||T_2||X||e_i) \) using its own master key \( MK_{CH_j} \) in order to retrieve the information \( ID_i, ID_{CH_j}, u, e_i, T_1, T_2 \) and \( X \) as \( D_{MK_{CH_j}}[E_{MK_{CH_j}}(ID_i||ID_{CH_j}||u||T_1||T_2||X||e_i)] = (ID_i||ID_{CH_j}||u||T_1||T_2||X||e_i) \). \( CH_j \) then checks if the retrieved \( ID_i \) is equal to the received \( ID_i \) and
also if the retrieved $ID_{CH_j}$ is equal to the received $ID_{CH_j}$. If these hold, $CH_j$ further checks if $|T_2 - T^*_2| \leq \Delta T_2$, where $T^*_2$ is the current system timestamp of the $CH_j$. If it holds good, $CH_j$ computes $v = e_i \oplus X = h(RPW_i||X_A)$, $w = h(v||T_2) = h(h(RPW_i||X_A)||T_2)$. $CH_j$ then checks if $w = u$. If it does not hold, the scheme terminates. Otherwise, if it holds, the user $U_i$ is considered as a valid user and authenticated by $CH_j$. After that $CH_j$ computes a secret session key $SK_{U_i,CH_j}$ shared with the user $U_i$ as $SK_{U_i,CH_j} = h(ID_i||ID_{CH_j}||e_i||T_1)$. Finally, $CH_j$ sends an acknowledgment to the user $U_i$ via other cluster heads and the BS and responds to the query of the user $U_i$.

- **Step 5:** After receiving the acknowledgment from $CH_j$, the user $U_i$ also computes the same secret session key shared with $CH_j$ using its previous system timestamp $T_1$, $ID_i$, $ID_{CH_j}$ and $e_i$ as $SK_{U_i,CH_j} = h(ID_i||ID_{CH_j}||e_i||T_1)$. Thus, both user $U_i$ and cluster head $CH_j$ will communicate securely in future using the derived secret session key $SK_{U_i,CH_j}$.

This authentication phase is summarized in Table 4.4.

**Remark 4.1:** This chapter concentrates on the user authentication problem in hierarchical WSNs. In our scheme, in order to access the real-time data from a cluster head $CH_j$ by a legal user $U_i$, it is very essential that the authentication between $U_i$ and $CH_j$ must take place. For this purpose, both $U_i$ and $CH_j$ must establish a secret session key $SK_{U_i,CH_j}$ for their future secure communication between them using this established session key $SK_{U_i,CH_j}$. Note that $CH_j$ can encrypt the messages for the user $U_i$ using the established session key $SK_{U_i,CH_j}$ and then send these encrypted messages to the user $U_i$ via its neighbor cluster heads or directly via the BS. In this case, the end-to-end encryption mechanism is used instead of the link-by-link encryption mechanism [130] in order to avoid a lot of computation overhead due to encryption/decryption at the intermediate nodes between $U_i$ and $CH_j$. On the other hand, the user $U_i$ can directly apply the decryption on the received encrypted messages from $CH_j$ using the same session key $SK_{U_i,CH_j}$. Thus, it is clear that the secure communication between $U_i$ and $CH_j$ take place directly, and it is the main goal of our proposed scheme in this thesis.
4.2 Description of the proposed scheme

Table 4.4: Authentication phase of our proposed scheme.

<table>
<thead>
<tr>
<th>BS</th>
<th>CH&lt;sub&gt;j&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Computes</td>
<td></td>
</tr>
<tr>
<td>( K = E_{MK_{CH_j}}(ID_i</td>
<td></td>
</tr>
<tr>
<td>Retrieves ( ID_i, ID_{CH_j}, N_i, )</td>
<td></td>
</tr>
<tr>
<td>( e_i, ) and ( T_1 ) using key ( K ).</td>
<td></td>
</tr>
<tr>
<td>Verifies ( ID_i, ID_{CH_j}, ) and (</td>
<td>T_1 - T_1^*</td>
</tr>
<tr>
<td>2. If all above hold, computes ( X = h(ID_i</td>
<td></td>
</tr>
<tr>
<td>If ( Z = N_i ), ( U_i ) is valid.</td>
<td></td>
</tr>
<tr>
<td>Computes ( u = h(Y</td>
<td></td>
</tr>
<tr>
<td>3. ( (ID_i</td>
<td></td>
</tr>
<tr>
<td>4. Decrypts ( E_{MK_{CH_j}}(ID_i</td>
<td></td>
</tr>
<tr>
<td>Checks if retrieved ( ID_i = ) received ( ID_i ) and retrieved ( ID_{CH_j} = ) received ( ID_{CH_j} ).</td>
<td></td>
</tr>
<tr>
<td>If these hold, checks if (</td>
<td>T_2 - T_2^*</td>
</tr>
<tr>
<td>If it holds, computes ( v = e_i \oplus X = h(RPW_i</td>
<td></td>
</tr>
<tr>
<td>5. Checks if ( w = u ).</td>
<td></td>
</tr>
<tr>
<td>If above holds, the user ( U_i ) is authenticated and responds to the query of the user ( U_i ).</td>
<td></td>
</tr>
<tr>
<td>Computes the secret session key ( SK_{U_i,CH_j} = h(ID_i</td>
<td></td>
</tr>
<tr>
<td>6. ( \langle \text{acknowledgment to BS} \rangle )</td>
<td></td>
</tr>
<tr>
<td>7. ( \langle \text{query response/data to user } U_i \rangle )</td>
<td></td>
</tr>
</tbody>
</table>
4.2.6 Password change phase

In this phase, any user $U_i$ can change his/her password freely and completely locally without the help of the $BS$. This phase contains the following steps:

- **Step 1:** $U_i$ inputs his/her smart card into the card reader of a specific terminal and provides his/her old password $PW_i^{old}$ as well as new changed password $PW_i^{new}$. After that the smart card computes the masked old password of the user $U_i$ as $RPW_i^* = h(y||PW_i^{old})$, $M_1 = h(RPW_i^*||X_A)$, and $M_2 = h(y||M_1)$.

- **Step 2:** The smart card then compares the computed value of $M_2$ with the stored $r_i$ in its memory. If they do not match, this means that the user $U_i$ has entered his/her old password $PW_i^{old}$ incorrectly and hence, the password change phase terminates immediately.

- **Step 3:** The smart card computes

  $$M_3 = e_i \oplus M_1 = h(ID_i||X_s),$$
  $$M_4 = h(y||PW_i^{new}),$$
  $$r_i' = h(y||M_4),$$
  $$M_5 = h(M_4||X_A),$$
  $$e_i' = M_3 \oplus M_5 = h(ID_i||X_s) \oplus h(y||PW_i^{new})||X_A).$$

- **Step 4:** Finally, the smart card replaces $r_i$ with $r_i'$ and $e_i$ with $e_i'$ into its memory.

4.2.7 Dynamic node addition phase

In this phase, we describe the method for adding new nodes in the existing network. If some sensor nodes or cluster heads are captured by an attacker or some nodes expire due to energy problem, we require to add some new nodes in the network. We consider the following two cases:

**Case I.** Addition of sensor nodes

In this case, if a sensor node $S_i$ is deployed in a cluster $C_i$, prior to deployment
of that node, the BS assigns the unique identifier $ID_{S_i}$ and also randomly generated unique master key $MK_{S_i}$ to it. These information are loaded in the memory of the node $S_i$.

**Case II.** Addition of cluster heads

When a cluster head $CH_j$ needs to be added in a cluster $C_i$ prior to deployment of that cluster head, $CH_j$, the BS assigns the unique identifier $ID_{CH_j}$ to it and also randomly generated unique master key $MK_{CH_j}$ to it as already done in registration phase (see Section 4.2.3). These information are finally loaded in the memory of the cluster head $CH_i$.

After deploying sensor nodes in their corresponding cluster along with its cluster head in the deployment field, the BS informs the user $U_i$ about the addition of the cluster head. Thus, it is noted that no other information is required to store in the user’s smart card regarding the addition of cluster head.

### 4.3 Security analysis of the proposed scheme

In this section, for security analysis of our proposed scheme, we use the threat model described in Section 4.1.2. We show that our scheme can resist against different known attacks, which are discussed in the following subsections.

#### 4.3.1 Replay attack

Suppose an attacker intercepts a valid login request message $\langle ID_i || ID_{CH_j} || E_{K_j}(ID_i || ID_{CH_j} || N_i || e_i || T_1) \rangle$ in the login phase and tries to login to the BS by replaying the same. The verification of this login request will fail in our scheme by the attacker because he/she has to know $N_i$, $e_i$ and $T_1$. However, to retrieve these information from the login request message, he/she needs to know the encrypted master key $E_{MK_{CH_j}}(ID_i || ID_{CH_j} || X_s)$ of the cluster head $CH_j$. The attacker is further unable to compute $X$, $Y$, $Z$ and $u$ for verification of the login request message. Thus, the proposed scheme can resist replay attack.
4.3.2 Many logged-in users with the same login-id attack

The proposed scheme can prevent the threat of the many logged-in users with the same login-id. The systems which maintain the password table to verify user login are usually vulnerable to this kind of attack. However, our scheme requires on-card computation for logging to the WSN, and once the smart card is removed from the system, the login process is aborted. Consider an example, where two users $U_i$ and $U_j$ have the same password. The random secret number $y$ is used in computation of their masked passwords. Hence, even if two users have same ids and passwords, they will have different the masked passwords. As a result, even if two users have same password, problem of many logged in users with same login id does not arise in our scheme, and hence, our scheme resists this kind of threat.

4.3.3 Stolen-verifier attack

Our scheme is resilient against stolen-verifier attack, because the proposed scheme does not require to store any verifier/password table for verification. It is also noted that the $BS$, cluster heads and sensor nodes do not keep password tables. As a result, in our scheme an attacker cannot steal users’ password tables.

4.3.4 Password guessing attack

In our scheme, the login message $\langle ID_i || ID_{CH_j} || E_{K_j}(ID_i || ID_{CH_j} || N_i || e_i || T_1) \rangle$ is transmitted. The attacker cannot guess the user’s password $PW_i$ from $E_{K_j}(ID_i || ID_{CH_j} || N_i || e_i || T_1)$ because the attacker has to decrypt it using the encrypted master key $K_j = E_{MK_{CH_j}}(ID_i || ID_{CH_j} || X_s)$ of the cluster head $CH_j$ and then to retrieve $N_i$ and $e_i$. Moreover, even if attacker knows the master key $MK_{CH_j}$ of the cluster head $CH_j$ after capturing that $CH_j$ in the network, he/she is still unable to get the encrypted master key $K_j = E_{MK_{CH_j}}(ID_i || ID_{CH_j} || X_s)$ of the cluster head $CH_j$ because the secret information $X_s$ is unknown to the attacker ($X_s$ is only known to the $BS$). Thus, computing user’s password $PW_i$ from $N_i$ and $e_i$ is computationally infeasible problem due to the one-way property of the hash function $h(\cdot)$. In this way, our scheme can resist password guessing attack.
4.3 Security analysis of the proposed scheme

4.3.5 Password change attack

Suppose a legal user lost his/her smart card or his/her smart card has been stolen by an attacker. Suppose the attacker can also breach the information \( \{ID_i, y, X_A, r_i, e_i, h(\cdot), \{(K_j, ID_{CH_j}) \mid 1 \leq j \leq m + m'\}\} \) which are stored in the smart card. In our scheme, for changing the password the attacker has to pass the old password \( PW_i^{\text{old}} \) verification. However, it is computationally infeasible to derive the old password \( PW_i^{\text{old}} \) from \( r_i \) and \( e_i \) due to the one-way property of the hash function \( h(\cdot) \) and the secret information \( X_s \) is only maintained by the BS. As a result, for a success in this attack the attacker has to guess the old password before updating the new password chosen by him/her.

4.3.6 Resilience against node capture attack

We measure the resilience against node capture attack of a user authentication scheme in WSN by estimating the fraction of total secure communications that are compromised by a capture of \( c \) nodes not including the communication in which the compromised nodes are directly involved, that is, we want to find out the effect of \( c \) cluster nodes being compromised on the rest of the network. For example, for any non-compromised cluster head \( CH_j \), we need to find out the probability that the adversary can decrypt the secure communication between \( CH_j \) and a user \( U_i \) when \( c \) cluster heads are already compromised. We denote this probability by \( P_e(c) \). Now, if \( P_e(c) = 0 \), we call such user authentication scheme as unconditionally secure against node capture attack or perfectly resilient against node capture attack.

Due to unattended nature of WSN, nodes can be captured by an attacker. Assume that some cluster heads are captured by the attacker. If the attacker captures a cluster head, he/she knows its master key from its memory. Note that each node (including cluster head) is given prior to its deployment in the target field a unique randomly generated master key. Thus, the attacker will be able to compromise the master key of that captured cluster head only. Using that compromised master key the attacker can response with false data to the legitimate user only. However, other non-compromised nodes can still communicate securely with the actual real-time data to the legitimate users. As a result, compromise of that captured cluster head does not lead to compromise any other secure communication between the user and the non-compromised nodes in the network. In this way, our scheme provides...
unconditional security against node capture attack.

**Remark 4.2:** When a cluster head (CH) is compromised, an attacker then compromises its own master key and session key shared with a user. In addition, secure communication with its neighbor sensor nodes are also compromised. However, other non-compromised sensor nodes in the cluster in which compromised CH is charge of, communicate securely with each other in that cluster because we use the unconditionally secure key management scheme [36] for secure communication between nodes in each cluster. In [36], we take 220 sensor nodes in a cluster so that any two neighbor nodes can establish a secure link for their secure communication when each sensor node and cluster head store 200 keys in their key rings. Thus, if we have \( m = 100 \) clusters in the sensor network which are initially deployed, then the total number of sensor nodes can be deployed is \( 220 \times 100 = 22000 \), which constitutes a large-scale WSN. According to threat model (given in Section 4.1.2), the compromised CHs and sensor nodes in each cluster can be detected and as a result, the BS knows about compromised nodes in the network. The BS then informs the user \( U_i \) about the compromised CHs and thus, \( U_i \) will not send any queries in order to retrieve data from the compromised CHs. Therefore, it is necessary to re-deploy new cluster heads or sensor nodes for the compromised CHs or sensor nodes in the existing network. We first consider that a sensor node is compromised by an attacker and it is replaced by a new sensor node. In our scheme, during dynamic sensor nodes addition phase (described in Section 4.2.7), a new sensor node is assigned its own identifier and a unique randomly generated master key which are different from the identifiers and master keys of the compromised nodes. After its deployment, it will establish with its neighbor sensor nodes as well as its cluster head (if the cluster head is its neighbor) in the cluster using [36]. Note that according to [36], each secure link uses a distinct secret pairwise key. We now consider that a cluster head in a cluster is compromised and that CH will be replaced by a new CH in that cluster. From our dynamic cluster heads addition phase (described in Section 4.2.7), we see that new cluster head is assigned its own identifier and a unique randomly generated master key before its deployment in the existing network, which are different from the identifiers and master keys of the compromised nodes. We further note that the user \( U_i \)’s smart card contains additional \( m' \) key-plus-identifier combinations \( \{(K_{m+j}, ID_{CH_{m+j}}) \mid 1 \leq j \leq m'\} \), and hence we can deploy additional \( m' \) cluster
heads $CH_{m+j}(1 \leq j \leq m')$ after initial deployment into the existing network. In our scheme, the new deployed cluster head needs to be one of the $m'$ cluster heads $CH_{m+j}$ ($1 \leq j \leq m'$) for which the information are already stored in $U_i$'s smart card. For example, a compromised cluster head $CH_j$ in a cluster will be replaced by a cluster head $CH_{m+j}$, for some $j \in \{1, 2, \ldots, m'\}$. After its deployment in that cluster, it establishes secure communication links with its neighbor sensor nodes in that cluster and also with its neighbor cluster heads in the network using [36]. As a result, there is no need to update information about addition of $CH_{m+j}$ in the user $U_i$'s smart card. Of course, our scheme tolerates addition of $m'$ (for example, $m' = 100$) new cluster heads in the network. In practice, we assume that not all $m$ cluster heads, which were deployed in the initial deployment, are compromised by the attacker, and hence, it is reasonable to believe that some cluster heads could be compromised by the attacker and they will be replaced by new cluster heads whose information are already in user’s smart card. Finally, the BS needs to inform $U_i$ about addition of cluster heads in clusters so that $U_i$ can access data from those cluster heads. Thus, our scheme is suitable when a cluster head is added into the existing network after initial deployment of nodes.

4.3.7 Smart card breach attack

An in Fan et al.'s scheme [68], although the smart card is assumed safe and cannot be cracked, however there is a risk of smart card crack. If an attacker/intruder attains a smart card and cracks it, he/she can obtain its stored information $ID_i$, $y$, $X_A$, $r_i = h(y||h(RPW_i||X_A))$, $e_i = f_i \oplus x = h(ID_i||X_s) \oplus h(RPW_i||X_A)$, $h(\cdot)$, $\{(K_j, ID_{CH_j}) \mid 1 \leq j \leq m + m'\}$, where $K_j = E_{MK_{CH_j}}(ID_i||ID_{CH_j}||X_s)$. However, the attacker has no feasible way to know the user $U_i$'s password $PW_i$ from $r_i$ and $e_i$ due to one-way property of the hash function $h(\cdot)$. Since the secret information $X_s$ is only known to the BS, there is no feasible way for the attacker to obtain the master key $MK_{CH_j}$ of the cluster head $CH_j$ from $K_j$ again due to one-way property of the hash function $h(\cdot)$. The attacker needs to guess the user $U_i$'s correct password $PW_i$ in order to pass the password verification in the login phase. Moreover, the computation of $N_i$ in the login phase becomes infeasible due to also one-way property of the hash function $h(\cdot)$. Hence, our scheme prevents from smart card breach attack.

Remark 4.3: According to our threat model (given in Section 4.1.2), the compro-
mised CHs and sensor nodes in each cluster can be detected and as a result, the BS knows about compromised nodes in the network. The BS then informs the user $U_i$ about compromised CH and thus $U_i$ will not send any queries in order to retrieve data from the compromised CH. Li et al. [97] uses a different threat model, where a cluster head $CH_j$ can steal a legal user $U_i$’s smart card. The smart card contains additional $m'$ key-plus-identifier combinations $\{(K_{m+j}, ID_{CH_{m+j}}) | 1 \leq j \leq m'\}$, and for additional $m'$ cluster heads $CH_{m+j}(1 \leq j \leq m')$ which are stored in $U_i$’s smart card can be extracted by launching power analysis attack [111]. In this case, that cluster head can use its own $MK_{CH_j}$ to decrypt $E_{MK_{CH_j}}(ID_i || ID_{CH_j} || X_s)$ and get the secret key $X_s$ of the base station. After getting the base station secret key $X_s$, the compromised cluster head $CH_j$ can reproduce a fake account for non-register user and the user clone attack can be launched. According to our threat model, this attack is not possible in practice, where a compromised cluster head can use a user’s stolen smart card. Li et. al. [98] suggested a solution to prevent the above attack without changing our authentication scheme. They proposed to use $E_{MK_{CH_j}}(ID_i || ID_{CH_j} || h(ID_i || ID_{CH_j} || X_s))$ in place of $E_{MK_{CH_j}}(ID_i || ID_{CH_j} || X_s)$. In that case secret key disclosure attack by a compromised cluster head is prevented as $X_s$ is hashed to $h(ID_i || ID_{CH_j} || X_s)$, due to one-way property of the hash function $h(\cdot)$.

4.3.8 Denial-of-service attack

In our scheme, the BS sends a message with expecting an acknowledgment from the cluster head requested by the user. The denial-of-service attack is not possible in our scheme because at the end of each user authentication, an acknowledgment is sent to the user $U_i$ via the BS which allows the user to duly know that response coming from cluster head is authentic.

4.3.9 Privileged-insider attack

During the registration phase of the proposed scheme (see in Section 4.2.3), the user $U_i$ does not send his/her password $PW_i$ in plaintext, instead the user $U_i$ sends the hashed password $RPW_i = h(y || PW_i)$ to the BS. It is computationally infeasible task to retrieve $PW_i$ from $RPW_i$ due to one-way property of the hash function $h(\cdot)$. The system manager or a privileged-insider of the BS does not know the password $PW_i$ of $U_i$, and he/she is thus unable to impersonate $U_i$ by accessing other
servers where \( U_i \) could be also a registered user if \( U_i \) uses the same password \( PW_i \) for his/her convenience of remembering long password and easy-to-use whenever required. Consequently, our scheme is secure against privileged-insider attack.

### 4.3.10 Masquerade attack

Consider that an illegal user may try to fabricate a fake login request message to cheat the \( BS \) to convince that it is a legal login request in the login phase. Note that the user sends the message \( \langle ID_i||ID_{CH_j}||E_{K_j}(ID_i||ID_{CH_j}||N_i||e_i||T_1) \rangle \) to the \( BS \). In order to convince the \( BS \) that it is a legal remote login request, the illegal user has to decrypt \( E_{K_j}(ID_i||ID_{CH_j}||N_i||e_i||T_1) \) using the key \( K_j \) correctly, where \( K_j = E_{MK_{CH_j}}(ID_i||ID_{CH_j}||X_s) \). Suppose the illegal user captures the cluster head \( CH_j \) in the network and gets its master key \( MK_{CH_j} \). Since the secret information \( X_s \) is only known to the \( BS \), even after knowing the master key \( MK_{CH_j} \) of the captured cluster head \( CH_j \), the illegal user cannot still compute the key \( K_j \) using the identifiers \( ID_i \) and \( ID_{CH_j} \). Thus, our scheme resists this attack.

### 4.3.11 Formal security analysis of the proposed scheme

In this section, through the formal security analysis, we show that our scheme has the ability to tolerate various known attacks. We show the formal security analysis of our scheme for protecting the secret session key and password guessing attack by an adversary. We follow the similar proof as in [20], [21], [42], [53], [112], [117], [118] for the formal security analysis of our proposed scheme. We have used the method of contradiction proof [25] for our formal security analysis. Note that one can also prove the formal security in the standard model. However, in this thesis, we have performed the formal security analysis under the generic group model of cryptography.

For the formal security analysis, we first define the formal definition of the indistinguishability of encryption under chosen plaintext attack (IND-CPA) as follows:

**Definition 4.1** (Indistinguishability of encryption under chosen plaintext attack (IND-CPA) [147]). The indistinguishability of encryption (IND) under chosen plaintext attack (CPA) is defined as follows. Let \( SE/ME \) be the single/multiple eavesdropper respectively, and \( O_{k_1}, O_{k_2}, \ldots, O_{k_N} \) be \( N \) different independent encryption
oracles associated with encryption keys $k_1, k_2, \ldots, k_N$ respectively. Define the advantage functions of $SE$ and $ME$, respectively, as $\text{Adv}_{\Omega,SE}^{\text{ind-cpa}}(l) = 2\Pr[SE \leftarrow O_{k_1}; (m_0, m_1 \leftarrow_R SE); \theta \leftarrow_R \{0, 1\}; \gamma \leftarrow_R O_{k_1}(m_\theta) : SE(\gamma) = \theta] - 1$, and $\text{Adv}_{\Omega,ME}^{\text{ind-cpa}}(l) = 2\Pr[ME \leftarrow O_{k_1}, \ldots, O_{k_N}; (m_0, m_1 \leftarrow_R ME); \theta \leftarrow_R \{0, 1\}; \gamma_1 \leftarrow_R O_{k_1}(m_\theta), \ldots, \gamma_N \leftarrow_R O_{k_N}(m_\theta) : ME(\gamma_1, \ldots, \gamma_N) = \theta] - 1$, where $\Omega$ is the encryption scheme and $\Pr[X]$ denotes the probability of an event $X$. Then the encryption scheme $\Omega$ is IND-CPA secure in the single (multiple) eavesdropper setting if $\text{Adv}_{\Omega,SE}^{\text{ind-cpa}}(l)$ (respectively, $\text{Adv}_{\Omega,ME}^{\text{ind-cpa}}(l)$) is negligible (in the security parameter $l$) for any probabilistic, polynomial time (PPT) adversary $SE$ ($ME$).

To apply the the method of contradiction proof [25] for our formal security analysis, we assume that the following two random oracles exist for an adversary, say $A$ as follows:

- **Reveal1**: This random oracle will unconditionally output the input $x$ from the corresponding encrypted value $E_k(x)$ without knowing the symmetric key $k$, where $E_k(\cdot)$ is the symmetric-key encryption using the key $k$.

- **Reveal2**: This random oracle will unconditionally output the input $x$ from the corresponding hash value $y = h(x)$, where $h(\cdot)$ is the one-way hash function.

**Theorem 4.1.** Let the used symmetric key encryption scheme, $\Omega$ be IND-CPA secure. Then our proposed scheme is secure against an adversary for compromising the secret session key between the user $U_i$ and a cluster head $CH_j$.

**Proof.** In this proof, we require to construct an adversary $A$ who has the ability to derive the secret session key between any user $U_i$ and any cluster head $CH_j$. We use the Reveal1 oracle for the adversary $A$ for this purpose. The adversary $A$ runs the experimental algorithm $\text{Exp}_1^{\text{IND-CPA}}_{\text{DPUAS},A}$ given in Algorithm 2 for our proposed dynamic password-based user authentication scheme, say, DPUAS. Note that the experiment $\text{Exp}_1^{\text{IND-CPA}}_{\text{DPUAS},A}$ given in Algorithm 2 means that an experiment, named Exp1, for our proposed dynamic password-based user authentication scheme, DPUAS, is run by the adversary $A$ and this experiment is based on the difficulty of solving IND-CPA problem.

We define the success probability for $\text{Exp}_1^{\text{IND-CPA}}_{\text{DPUAS},A}$ provided in Algorithm 2 as $\text{Succ}_1^{\text{IND-CPA}}_{\text{DPUAS},A} = \Pr[\text{Exp}_1^{\text{IND-CPA}}_{\text{DPUAS},A} = 1] - 1$, where $\Pr[E]$ denotes the probability of
4.3 Security analysis of the proposed scheme

Algorithm 2 $\text{Exp}^{\text{IND-CPA}}_{\text{DPUAS,A}}$

1: Intercept the message $(ID_i||ID_{CH_j}||E_{MK_{CH_j}}(ID_i||ID_{CH_j}||u||T_1||T_2||X||e_i))$ during the authentication phase, which is sent from base station (BS) to a sensor node $SN_i$.

2: Call $\text{Reveal}_1$ oracle on the input $E_{MK_{CH_j}}(ID_i||ID_{CH_j}||u||T_1||T_2||X||e_i)$ to retrieve the information $u,T_1,T_2,X$ and $e_i$. Let $(ID_i'||ID_{CH_j}'||u'||T_1'||T_2'||X'||e_i') \leftarrow \text{Reveal}_1(E_{MK_{CH_j}}(ID_i||ID_{CH_j}||u||T_1||T_2||X||e_i))$.

3: Compute $v' = e_i' \oplus X'$ using the retrieved values $e_i'$ and $X'$ in Step 2.

4: Compute the hash value $w' = h(v'||T_2')$ using the retrieved value $T_2'$ in Step 2 and computed value $v'$ in Step 3.

5: if $w' = u$ then
6: Compute the secret key $SK_{U_i,CH_j} = h(ID_i||ID_{CH_j}||e_i'||T_1')$ and accept it as the correct key.
7: return 1 (Success)
8: else
9: return 0 (Failure)
10: end if

an event $E$. The advantage function for $\text{Exp}^{\text{IND-CPA}}_{\text{DPUAS,A}}$ then becomes $\text{Adv}^{\text{IND-CPA}}_{\text{DPUAS,A}}(t_1,q_{R_1}) = \max_{\mathcal{A}}\{\text{Succ}^{\text{IND-CPA}}_{\text{DPUAS,A}}\}$, where the maximum is taken over all $\mathcal{A}$ with execution time $t_1$ and the number of queries $q_{R_1}$ made to the $\text{Reveal}_1$ oracle. We call our scheme is provably secure against an adversary $\mathcal{A}$ for deriving the secret symmetric key $SK_{U_i,CH_j}$ between the user $U_i$ and cluster head $CH_j$ and as a result, our scheme prevents against compromising the secret session key between the user $U_i$ and a cluster head $CH_j$ by an adversary, if $\text{Adv}^{\text{IND-CPA}}_{\text{DPUAS}}(t_1,q_{R_1}) \leq \epsilon$, for any sufficiently small $\epsilon > 0$.

Consider the experiment $\text{Exp}^{\text{IND-CPA}}_{\text{DPUAS,A}}$ given in Algorithm 2. If $\mathcal{A}$ has the ability to solve IND-CPA, then the adversary $\mathcal{A}$ can correctly compute the secret shared key $SK_{U_i,CH_j}$ between the user $U_i$ and and a cluster head $CH_j$, and win the game, using the $\text{Reveal}_1$ oracle. Once the secret shared key $SK_{U_i,CH_j}$ is known to the adversary $\mathcal{A}$, he/she can eavesdrop or intercept any message transmitted between the user $U_i$ and the cluster head $CH_j$, and as a result, he/she can decrypt all the transmitted messages between $U_i$ and $CH_j$, which use the session key $SK_{U_i,CH_j}$. However, it is a contradiction, because from Definition 4.1 that it is a hard problem for the
Theorem 4.2. Let the used symmetric key encryption scheme, \( \Omega \) be IND-CPA.
4.4 Formal security verification of our scheme using AVISPA tool

secure. Then our proposed scheme is secure against an adversary for deriving the password $PW_i$ of a user $U_i$, if the one-way hash function $h(\cdot)$ behaves like a random oracle.

\textit{Proof.} In this proof, we require to construct an adversary $\mathcal{A}$ so that he/she can claim as a genuine user $U_i$ and can guess the password $PW_i$ a user $U_i$ and thus, he/she can be successful to mount password guessing attack. In order to do so, the adversary $\mathcal{A}$ runs the experimental algorithm $Exp_{\text{IND-CPA,\text{HASH}}}^{2,DPUAS,\mathcal{A}}$ given in Algorithm 3 for our proposed dynamic password-based user authentication scheme, $DPUAS$.

We define the success probability for $Exp_{\text{IND-CPA,\text{HASH}}}^{2,DPUAS,\mathcal{A}}$ provided in Algorithm 3 as $\text{Succ}_{\text{IND-CPA,\text{HASH}}}^{2,DPUAS,\mathcal{A}} = Pr[Exp_{\text{IND-CPA,\text{HASH}}}^{2,DPUAS,\mathcal{A}} = 1] - 1$ and the advantage function for this experiment, $Exp_{\text{IND-CPA,\text{HASH}}}^{2,DPUAS,\mathcal{A}}$ becomes $\text{Adv}_{\text{IND-CPA,\text{HASH}}}^{2,DPUAS}(t_2, q_{R_2}) = \max_{\mathcal{A}}\{\text{Succ}_{\text{IND-CPA,\text{HASH}}}^{2,DPUAS,\mathcal{A}}\}$, where the maximum is taken over all $\mathcal{A}$ with execution time $t_2$ and $q_{R_2}$ is the number of queries made to the $\text{Reveal}_2$ oracle.

According to the experiment, the adversary $\mathcal{A}$ can derive the password $PW_i$ of the user $U_i$ and win the game, using both $\text{Reveal}_1$ and $\text{Reveal}_2$ oracles. However, it is a contradiction, because deriving the input from a given hash value is a computationally infeasible task as the hash function $h(\cdot)$ closely behaves like a random oracle given in Definition 2.1 and IND-CPA given in Definition 4.1 is provably secure. Thus, $\text{Adv}_{\text{IND-CPA,\text{HASH}}}^{2,DPUAS}(t_2, q_{R_2}) \leq \epsilon$, for any sufficiently small $\epsilon > 0$ as it is dependent on both $\text{Adv}_A^{\text{HASH}}(t)$ and $\text{Adv}_{\Omega,SE}^{1,\text{IND-CPA}}(l)$ (respectively $\text{Adv}_{\Omega,ME}^{1,\text{IND-CPA}}(l)$). Hence, our scheme is secure against an adversary for deriving the password of any user.

\begin{flushright}
\hfill $\square$
\end{flushright}

\section{4.4 Formal security verification of our scheme using AVISPA tool}

In this section, we first describe in brief the overview of AVISPA (Automated Valida-
tion of Internet Security Protocols and Applications) tool along with the high-level protocol specification language (HLPSL). We then give the implementation details of our scheme in HLPSL. Finally, we discuss the analysis of the simulation results using AVISPA back-end.
4.4.1 Overview of AVISPA

AVISPA is a push-button tool for the automated validation of Internet security-sensitive protocols and applications, which provides a modular and expressive formal language for specifying protocols and their security properties, and integrates different back-ends that implement a variety of state-of-the-art automatic analysis techniques [4], [67]. The architecture of the AVISPA tool is shown in Figure 4.3 [3]. We have used the widely-accepted AVISPA back-ends for our formal security verification [21], [42], [47], [52], [81], [108], [112]. AVISPA implements four back-ends and abstraction-based methods which are integrated through the high level protocol specific language, known as HLPSL [137]. A static analysis is performed to check the executability of the protocol, and then the protocol and the intruder actions are compiled into an intermediate format (IF). The intermediate format is the start point for the four automated protocol analysis techniques. IF is a lower-level language than HLPSL and is read directly by the back-ends to the AVISPA tool. The IF describes a protocol in terms of rewrite rules describing an infinite-state transition system with an initial state, transition rules, and a state-based safety property, namely a goal (attack) predicate that defines if a given state is an attack state or not. As described in [4], a formal semantics reflects the interpretation of the IF formally. For example, formally, we compose (or compress) several steps: when the intruder sends a message, an agent reacts to it according to his rules, and the intruder diverts immediately the agent’s answer [4]. The first back-end, the On-the-fly Model-Checker (OFMC), does several symbolic techniques to explore the state space in a demand-driven way [14]. The second back-end, the CL-AtSe (Constraint-Logic-based Attack Searcher), provides a translation from any security protocol specification written as transition relation in intermediate format into a set of constraints which are effectively used to find whether there are attacks on protocols. The third back-end, the SAT-based Model-Checker (SATMC), builds a propositional formula which is then fed to a state-of-the-art SAT solver and any model found is translated back into an attack. Finally, the fourth back-end, TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols), approximates the intruder knowledge by using regular tree languages.

Protocols to be implemented by the AVISPA tool have to be specified in HLPSL (High Level Protocols Specification Language) [137], and written in a file with extension hlpsl. This language is based on roles: basic roles for representing each par-
4.4 Formal security verification of our scheme using AVISPA tool

The participant role, and composition roles for representing scenarios of basic roles. Each role is independent from the others, getting some initial information by parameters, communicating with the other roles by channels. The intruder is modeled using the Dolev-Yao model [61] (as in the threat model used in our scheme) with the possibility for the intruder to assume a legitimate role in a protocol run. The role system also defines the number of sessions, the number of principals and the roles.

The output format (OF) of AVISPA is generated by using one of the four back-ends explained above. When the analysis of a protocol has been successful (by finding an attack or not), the output describes precisely what is the result, and under what conditions it has been obtained. In OF, there are the following sections.

- The first printed section SUMMARY indicates that whether the tested protocol is safe, unsafe, or whether the analysis is inconclusive.
- The second section, called DETAILS either explains under what condition the tested protocol is declared safe, or what conditions have been used for finding an attack, or finally why the analysis was inconclusive.
- Other sections such as PROTOCOL, GOAL and BACKEND are the name...
of the protocol, the goal of the analysis and the name of the back-end used, respectively.

- Finally, after some comments and statistics, the trace of an attack (if any) is also printed in the standard Alice-Bob format.

### 4.4.2 Specifying the protocol

We have implemented our proposed scheme in HLPSL language. In our implementation, we have three basic roles: alice, bs and bob, which represent the participants: the user \( U_i \), the base station \( BS \) and the cluster head \( CH_j \), respectively. We have further specified the session and environment in our implementation.

In Figure 4.4, we have implemented the role for the user \( U_i \) in HLPSL. During the registration phase, the user \( U_i \) sends the registration request message \( \langle ID_i||RPW_i \rangle \) securely to the BS with the help of the \( \text{Snd}() \) operation. Here the type declaration \( \text{channel} \) \((\text{dy})\) indicates that the channel is for the Dolev-Yao threat model. \( U_i \) waits for the smart card containing the information in the message \( \langle ID_i||y||X_A||r_i||e_i||h(\cdot),\{(K_j||ID_{CH_j}) | 1 \leq j \leq m + m'\} \rangle \) securely from the BS from the \( \text{Rcv}() \) operation. The intruder has the ability to intercept, analyze, and/or modify messages transmitted over the insecure channel. In the login phase, the user \( U_i \) sends the login request message \( \langle ID_i||ID_{CH_j},E_{K_j}(ID_i||ID_{CH_j},||N_i||e_i||T_1) \rangle \) to the BS. The user then waits for an acknowledgment regarding successful authentication from the cluster head \( CH_j \) via the BS. The basic types available in HLPSL are [4]:

- **agent**: Values of type *agent* represent principal names. The intruder is always assumed to have the special identifier \( i \).

- **public_key**: These values represent agents’ public keys in a public-key cryptosystem. For example, given a public (respectively private) key \( pk \), its inverse private (respectively public) key is obtained by \( \text{inv} \_pk \).

- **symmetric_key**: Variables of this type represent keys for a symmetric-key cryptosystem.

- **text**: In HLPSL, *text* values are often used as nonces. These values can be used for messages. If \( Na \) is of type *text* (fresh), then \( Na' \) will be a fresh value which the intruder cannot guess.
4.4 Formal security verification of our scheme using AVISPA tool

role alice (Ui, CHj, BS : agent,
            MKchj : symmetric_key,
            SKubs : symmetric_key,
            H : hash_func,
            Snd, Rcv: channel(dy))

played_by Ui
def=
local State : nat,
    RPWi, PWi, Xs, Xa, Yy, ACK,
    T1, T2, IDi, IDchj : text

Const alice_server, server_bob, subs1,
    subs2, subs3, subs4, subs5 : protocol_id

init State := 0

transition
1. State = 0 \& Rcv(start) =>
   State' := 1 \& RPWi' := H(Yy.PWi)
   \& Snd(Ui.BS.{IDi.RPWi'}_SKubs)
   \& secret({Xs}, subs1, BS)
   \& secret({PWi}, subs2, Ui)
   \& secret({MKchj}, subs3, {BS.CHj})
   \& secret({SKubs}, subs4, {Ui,BS})
   \& secret({Xa,Yy}, subs5, {Ui,BS})

                                      xor(H(IDi.Xs),H(H(H(Yy.PWi),Xa).H.
                                      [IDi.IDchj.Xs]_MKchj.IDchj]_SKubs) =>
   State' := 2 \& T1' := new()
   \& Snd(Ui.BS.IDi.IDchj.
          [IDi.IDchj.H(H(H(Yy.PWi).Xa).T1')
          .xor(H(IDi.Xs),H(H(Yy.PWi),Xa)).
          T1'].([IDi.IDchj.Xs]_MKchj))
   \& witness(Ui, BS, alice_server, T1')

3. State = 2 \& Rcv(BS.Ui.ACK) =>
   State' := 3
end role

Figure 4.4: Role specification in HLPSL for the user Ui.

- **nat**: The nat type represents the natural numbers in non-message contexts.

- **const**: This type represents constants.

- **hash_func**: The base type hash_func represents cryptographic hash functions. The base type function also represents functions on the space of messages. It is assumed that the intruder cannot invert hash functions (in essence, that they are one-way).
• **bool**: Boolean values are useful for modeling, for instance, binary flags.

The space of legal messages are defined as the closure of the basic types. For a given message \(Msg\) and encryption key \(Key\), we denote \(\{Msg\}_Key\) as the symmetric/public-key encryption. The associative “\(\cdot\)” operator is used for concatenations.

The “\(played_\text{by}\ A\)” declaration indicates that the agent named in variable \(A\) will play in the role. A knowledge declaration (generally in the top-level Environment role) is used to specify the intruder’s initial knowledge. Immediate reaction transitions have the form \(X > Y\), which relate an event \(X\) and an action \(Y\). This means that whenever we take a transition that is labeled in such a way as to make the event predicate \(X\) true, we must immediately (that is, simultaneously) execute action \(Y\). If a variable \(V\) remains permanently secret, it is expressed by the goal secrecy_of \(V\). If \(V\) is ever obtained or derived by the intruder, a security violation will result.

Figure 4.5 shows the implementation for the role of the BS in HLPSL. During the registration phase, after receiving the message \(\langle ID_i || RPW_i \rangle\) securely from the user \(U_i\) from the Rcv() operation, the BS sends a smart card containing the information in the message \(\langle ID_i || ID_{CH_j} || E_{MKCH_j}(ID_i || ID_{CH_j} || N_i || e_i || T_1) \rangle\) securely to the user \(U_i\). In the login phase, the BS receives the login request message \(\langle ID_i || ID_{CH_j} || E_{Kj}(ID_i || ID_{CH_j} || N_i || e_i || T_1) \rangle\) from the user \(U_i\). During the authentication phase, the BS sends the authentication request message \(\langle ID_i || ID_{CH_j} || E_{MKCH_j}(ID_i || ID_{CH_j} || a || T_1 || T_2 || X || e_i) \rangle\) to the cluster head \(CH_j\). BS then waits for an acknowledgment regarding successful authentication from the cluster head \(CH_j\).

In Figure 4.6, we have implemented the the role of the cluster head \(CH_j\) in HLPSL. During the authentication phase, after receiving the authentication request message \(\langle ID_i || ID_{CH_j} || E_{MKCH_j}(ID_i || ID_{CH_j} || a || T_1 || T_2 || X || e_i) \rangle\) from the BS, the cluster head \(CH_j\) sends an acknowledgment regarding successful authentication to the BS. In HLPSL specification, witness(A,B,id,E) declares for a (weak) authentication property of \(A\) by \(B\) on \(E\), declares that agent \(A\) is witness for the information \(E\); this goal will be identified by the constant \(id\) in the goal section. request(B,A,id,E) means for a strong authentication property of \(A\) by \(B\) on \(E\), declares that agent \(B\) requests a check of the value \(E\); this goal will be identified by the constant \(id\) in the goal section. The intruder is always denoted by \(i\).

We have given the specifications in HLPSL for the role of goal, environment and session in Figures 4.7 and 4.8. In the session segment, all the basic roles: alice, bs and
4.4 Formal security verification of our scheme using AVISPA tool

Figure 4.5: Role specification in HLPSL for the base station, BS.

bob are instanced with concrete arguments. The top-level role (environment) defines in the specification of HLPSL, which contains the global constants and a composition of one or more sessions, where the intruder may play some roles as legitimate users. The intruder also participates in the execution of protocol as a concrete session. The current version of HLPSL supports the standard authentication and secrecy goals.
In our implementation, the following five secrecy goals and two authentications are verified:

- **secrecy** _of subs1_: It represents that $X_s$ is kept secret to the BS.

- **secrecy** _of subs2_: It represents that $PW_i$ is kept secret to the user $U_i$.

- **secrecy** _of subs3_: It represents that $MK_{CH_j}$ is secret to BS and the cluster head $CH_j$.

- **secrecy** _of subs4_: It represents that $SKubs$ is secret to $U_i$ and BS.

- **secrecy** _of subs5_: It represents that $X_a$ and $y$ are secret to $U_i$ and BS.

- **authentication on alice_server**: $U_i$ generates a random timestamp $T_1$, where $T_1$ is only known to $U_i$. If the BS gets $T_1$ from the message from $U_i$, the BS authenticates $U_i$ on $T_1$. 
4.4 Formal security verification of our scheme using AVISPA tool

role environment()
def=
  const ui, chj, bs: agent,
    mkchj: symmetric_key,
    skubs : symmetric_key,
    h : hash_func,
    ack, pwi, xs, xa, yy, t1, t2,
    idi, idchj: text,
    alice_server, server_bob,
    subs1, subs2, subs3, subs4, subs5 : protocol_id
intruder_knowledge = {ui, chj, bs, idi, idchj, h}
composition
  session(ui, chj, bs, mkchj, skubs, h)
\session(ui, chj, bs, mkchj, skubs, h)
\session(ui, chj, bs, mkchj, skubs, h)
end role

goal
  secrecy_of subs1
  secrecy_of subs2
  secrecy_of subs3
  secrecy_of subs4
  secrecy_of subs5
  authentication_on alice_server
  authentication_on server_bob
end goal
environment()

Figure 4.7: Role specification in HLPSL for the environment and goal.

- **authentication_on server_bob**: BS generates a random timestamp $T_1$, where $T_2$ is only known to BS. If the cluster head $CH_j$ receives $T_2$ from the message from the BS, $CH_j$ authenticates BS on $T_2$.

In the goal section of the protocol, we write

```hlpsl
authentication_on alice_server
authentication_on server_bob
```

to indicate that the witness and request goal facts containing those two protocol ids, alice_server and server_bob, should be taken into account.

### 4.4.3 Analysis of results

The On-the-Fly Model-Checker OFMC builds the infinite tree defined by the protocol analysis problem in a demand-driven way, i.e. on-the-fly, hence the name of
the back-end. This backend uses a number of symbolic techniques in order to represent the state-space. OFMC can be employed not only for efficient falsification of protocols (i.e. fast detection of attacks), but also for verification (i.e. proving the protocol correct) for a bounded number of sessions - without bounding the messages an intruder can generate [4].

We have chosen the back-end OFMC for an execution test and a bounded number of sessions model checking [14]. For the replay attack checking, the back-end checks whether the legitimate agents can execute the specified protocol by performing a search of a passive intruder. After that the back-end gives the intruder the knowledge of some normal sessions between the legitimate agents. For the Dolev-Yao model check, the back-end checks whether there is any man-in-the-middle attack possible by the intruder.

```
role session(Ui, CHj, BS: agent,
    MKchj : symmetric_key,
    SKubs : symmetric_key,
    H : hash_func )
  def=
    local SI, SJ, RI, RJ, BI, BJ: channel (dy)
  composition
    alice(Ui, CHj, BS, MKchj, SKubs, H, SI, RI)
    ∧ bs(Ui, CHj, BS, MKchj, SKubs, H, BI, BJ)
    ∧ bob(Ui, CHj, BS, MKchj, SKubs, H, SJ, RJ)
  end role
```

Figure 4.8: Role specification in HLPSL for the session.

We have simulated our proposed scheme under the back-ends: OFMC and CL-AtSe, using the AVISPA web tool under SPAN (Security Protocol ANimator for AVISPA) [13]. The simulation results are shown in Figures 4.9 and 4.10. The formal security verification analysis clearly shows that our scheme is secure against active attacks including replay and man-in-the-middle attacks.

### 4.5 Comparison with related schemes

In this section, we compare the performances of our proposed scheme with the existing related password-based schemes: Watro et al.’s scheme [141], Wong et al.’s
4.5 Comparison with related schemes

Figure 4.9: The result of the analysis using OFMC back-end.

Figure 4.10: The result of the analysis using CL-AtSe back-end.

scheme [144], M. L. Das’s scheme [58], Nyang-Lee’s scheme [116], Huang et al.’s scheme [79], He et al.’s scheme [73], Vaidya et al.’s scheme [134], Fan et al.’s scheme
We have compared the results of the formal security verification of all schemes in Table 4.5, under AVISPA model checker, OFMC. From this table, it is clear that Watro et al.'s scheme, M. L. Das’s scheme, Nyang-Lee’s scheme, He et al.’s scheme, Fan et al.’s scheme, Chen-Shih’s scheme and our scheme are secure against passive and active adversaries, while Wong et al.’s scheme, Huang et al.’s scheme and Vaidya et al.’s scheme are insecure against passive and active attacks including the replay and man-in-the-middle attacks.

Table 4.5: Simulation results of formal security verification among different schemes using AVISPA tool.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Results using OFMC and CL-AtSe backends</th>
</tr>
</thead>
<tbody>
<tr>
<td>Watro et al. [141]</td>
<td>Safe</td>
</tr>
<tr>
<td>Wong et al. [144]</td>
<td>Unsafe</td>
</tr>
<tr>
<td>M. L. Das [58]</td>
<td>Safe</td>
</tr>
<tr>
<td>Nyang-Lee [116]</td>
<td>Safe</td>
</tr>
<tr>
<td>Huang et al. [79]</td>
<td>Unsafe</td>
</tr>
<tr>
<td>He et al. [73]</td>
<td>Safe</td>
</tr>
<tr>
<td>Vaidya et al. [134]</td>
<td>Unsafe</td>
</tr>
<tr>
<td>Fan et al. [68]</td>
<td>Safe</td>
</tr>
<tr>
<td>Chen-Shih [23]</td>
<td>Safe</td>
</tr>
<tr>
<td>Ours</td>
<td>Safe</td>
</tr>
</tbody>
</table>

In Table 4.6, we have compared the secrecy level and functionality provided by [23], [58], [68], [73], [79], [116], [134], [141], [144] with that provided by our scheme. Password change feature is supported by our scheme, [73], [79] and [134], while our scheme, [79] and [134] are also resilient against node capture attack. Our scheme is secure against denial-of-service attack and also provides mutual authentication between the user and the cluster head. Among all the schemes, only our scheme and [68] establish a secret session key between the user and cluster head that allows the user to collect data from a particular cluster (data from a particular region of the target field) for a given session. In addition, our scheme prevents smart card breach attack as in Fan et al.’s scheme [68] and privileged-insider attack. Considering other
Table 4.6: Performance comparison between the proposed scheme and other related schemes.

<table>
<thead>
<tr>
<th></th>
<th>[141]</th>
<th>[144]</th>
<th>[58]</th>
<th>[116]</th>
<th>[79]</th>
<th>[73]</th>
<th>[134]</th>
<th>[68]</th>
<th>[23]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>$I_2$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>$I_3$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>$I_4$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>$I_5$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>$I_6$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>$I_7$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>$I_8$</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

$I_1$: whether supports change password or not; $I_2$: whether supports mutual authentication or not; $I_3$: whether resists denial-of-service attack or not; $I_4$: whether resilient against node capture attack or not; $I_5$: whether establishes secret session key between user and sensor node/cluster head or not; and $I_6$: whether establishes dynamic node addition or not; $I_7$: whether provides formal security analysis using random oracle models or not; $I_8$: whether secure under formal security verification using AVIPSA or not; Yes: the scheme is secure or supports the specified feature; No: the scheme is not secure or does not support the specified feature.

existing approaches, our scheme is better in terms of security. Besides these security aspects, our scheme supports addition of new cluster heads and sensor nodes easily in the existing network in order to replace the compromised cluster heads and sensor nodes. The proposed scheme does not require to update extra information about new nodes addition in the user’s smart card. In addition, we have shown that our scheme is provably secure under the random oracle models and secure against passive and active adversaries using AVISPA tool.

In Table 4.7, we have compared computational costs in different phases like registration, login and authentication phases of [58], [79], [116], [141], [144] with our scheme. We have ignored XOR operation in comparison, because XOR operation is negligible. Since the cluster head $CH_j$ and $BS$ are more resource-rich as compared to usual sensor nodes, our scheme makes advantages of using the computational power
of BS and cluster heads to provide secure log-in to user. This makes our scheme more efficient in terms of resource usage as compared to other related schemes.

In Table 4.8, we have compared the communication costs among our scheme and the related other schemes. The communication cost is considered in terms of number of message exchanges for a successful user authentication. From this table, it is noted that a successful user authentication process in our scheme requires four message exchanges, while Watro et al.’s scheme, Wong et al.’s scheme, M. L. Das’s scheme, Nyang-Lee’s scheme, Huang et al.’s scheme, He et al.’s scheme, Vaidya et al.’s scheme, Fan et al.’s scheme and Chen-Shih’s scheme require two, four, three, three, three, five, five and four message exchanges, respectively. Although Watro et al.’s scheme requires minimum number of message exchanges among all the schemes, their scheme is computational expensive due to involvement of public-key computations such as modular exponentiation operations in resource-constrained WSN. In our scheme, the message transmission between the BS and the cluster head is done effectively because the message transmission is done using a very few number of hops due to involvement of cluster heads in the communication path. In all other schemes, the message transmission between the BS (GW-node) and sensor node often requires multi-hop communication path and as a result our scheme is significantly efficient in term of communication cost as compared to the other schemes.

Finally, we compare the sensor node’s energy cost between the proposed scheme and the other schemes in Table 4.9. The sensor node’s energy cost is due to both computational and communication costs involved in the schemes. For Watro et al.’s scheme, a sensor node consumes battery due to nonce validation, checksum generation and verification, two public-key operations and then response to the user’s query. In Wong et al.’s scheme, a sensor node consumes battery for a lookup table query, three hash operations for parameters generation and then waiting for the GW-node’s response before responding to the user’s query. In M. L. Das’s scheme, a sensor node consumes battery due to timestamp validation and one hash operation for parameter generation and for responding to the user’s query. Battery power consumption for a sensor node in Nyang-Lee’s scheme is due to timestamp validation, one key derivation function (kdf) operation for computing encryption key, one kdf operation for computing MAC key, two MAC operations for parameter generation, one encryption for encrypting data and then another decryption for retrieving
Table 4.7: Comparison of computational costs in different phases between the proposed scheme and the other schemes.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>$U_i$</td>
<td>$t_{pu}$</td>
<td>$-t_{pr}$</td>
<td>$t_{pu}$</td>
<td>$3t_h$</td>
<td>$3t_h$</td>
<td>$4t_h$</td>
<td>$5t_h$</td>
<td>$6t_h$</td>
<td>$3t_h$</td>
<td>$(m+m')t_{enc}$+$3t_h$</td>
</tr>
<tr>
<td></td>
<td>BS</td>
<td>$t_{pr}$</td>
<td>$3t_h$</td>
<td>$3t_h$</td>
<td>$4t_h$</td>
<td>$4t_h$</td>
<td>$6t_h$</td>
<td>$3t_h$</td>
<td>$-t_{enc}$</td>
<td>$-t_{enc}$</td>
<td>$-t_{enc}$</td>
</tr>
<tr>
<td></td>
<td>$S_j$</td>
<td>$-t_{pr}$</td>
<td>$-t_{h}$</td>
<td>$4t_h$</td>
<td>$5t_h$</td>
<td>$6t_h$</td>
<td>$7t_h$</td>
<td>$4t_h$</td>
<td>$5t_h$</td>
<td>$t_{enc}$</td>
<td>$5t_h+t_{dec}$</td>
</tr>
<tr>
<td></td>
<td>$CH_j$</td>
<td>$-t_{pr}$</td>
<td>$-t_{h}$</td>
<td>$4t_h$</td>
<td>$5t_h$</td>
<td>$6t_h$</td>
<td>$7t_h$</td>
<td>$4t_h$</td>
<td>$5t_h$</td>
<td>$t_{dec}$</td>
<td>$3t_h+t_{enc}$</td>
</tr>
<tr>
<td>L</td>
<td>$U_i$</td>
<td>$2t_{pu}$</td>
<td>$-t_{pr}$</td>
<td>$4t_h$</td>
<td>$4t_h+t_{kdf}$</td>
<td>$2t_{enc}$</td>
<td>$4t_h$</td>
<td>$5t_h$</td>
<td>$6t_h$</td>
<td>$7t_h$</td>
<td>$4t_h$ $5t_h+t_{dec}$</td>
</tr>
<tr>
<td></td>
<td>+ BS</td>
<td>$-t_{pr}$</td>
<td>$4t_h$</td>
<td>$3t_h+t_{kdf}$</td>
<td>$2t_{enc}$</td>
<td>$5t_h$</td>
<td>$5t_h$</td>
<td>$2t_h$</td>
<td>$5t_h$</td>
<td>$3t_h+t_{dec}$</td>
<td>$2t_{enc}$</td>
</tr>
<tr>
<td></td>
<td>$S_j$</td>
<td>$2t_{pu}$</td>
<td>$3t_h$</td>
<td>$t_{h}$</td>
<td>$2t_{kdf}$</td>
<td>$t_{h}$</td>
<td>$2t_h$</td>
<td>$t_{h}$</td>
<td>$-t_{dec}$</td>
<td>$-t_{dec}$</td>
<td>$-t_{dec}$</td>
</tr>
<tr>
<td></td>
<td>$CH_j$</td>
<td>$-t_{pu}$</td>
<td>$-t_{h}$</td>
<td>$-t_{h}$</td>
<td>$-t_{h}$</td>
<td>$-t_{h}$</td>
<td>$8t_h$</td>
<td>$-t_{h}$</td>
<td>$2t_h$</td>
<td>$t_{dec}$</td>
<td>$2t_h+t_{dec}$</td>
</tr>
</tbody>
</table>
Table 4.8: Comparison of communication costs between the proposed scheme and the other schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Communication cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Watro et al. [141]</td>
<td>2 messages</td>
</tr>
<tr>
<td>Wong et al. [144]</td>
<td>4 messages</td>
</tr>
<tr>
<td>M. L. Das [58]</td>
<td>3 messages</td>
</tr>
<tr>
<td>Nyang-Lee [116]</td>
<td>3 messages</td>
</tr>
<tr>
<td>Huang et al. [79]</td>
<td>3 messages</td>
</tr>
<tr>
<td>He et al. [73]</td>
<td>3 messages</td>
</tr>
<tr>
<td>Vaidya et al. [134]</td>
<td>5 messages</td>
</tr>
<tr>
<td>Fan et al. [68]</td>
<td>5 messages</td>
</tr>
<tr>
<td>Chen-Shih [23]</td>
<td>4 messages</td>
</tr>
<tr>
<td>Ours</td>
<td>4 messages</td>
</tr>
</tbody>
</table>

MAC and encryption keys shared between the user and the sensor node. In case of Huang et al.’s scheme, a sensor node consumes battery due to timestamp validation and one hash operation for parameter verification and finally for responding to the user’s query. In He et al.’s scheme, a sensor node consumes battery due to timestamp validation, one hash function for parameter generation and response to the user’s query. In case of Vaidya et al.’s scheme, battery consumption for a sensor node comes due to timestamp validation, one hash function for parameter generation, another hash function for parameter verification, and then response to the user’s query and waiting for the GW-node’s response. Fan et al’s scheme requires battery consumption for a sensor node due to one hash function for random-nonce validation, another hash function for session key generation and then response to the user’s query. Chen-Shih’s scheme needs battery consumption for a sensor node due to time-stamp validation, one hash function for parameter generation and response to the user’s query. Finally in our scheme, a cluster head consumes battery due to timestamp validation, two hash operations for parameter generation and session key generation, one symmetric key decryption and response to the user’s query and sending an acknowledgment to the user via the BS for a successful user authentication and session key establishment. Note that in our scheme, a sensor node does
### 4.5 Comparison with related schemes

Table 4.9: Comparison of energy cost of sensor node/cluster head between the proposed scheme and the other schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Sensor node/cluster head’s energy cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>[141]</td>
<td>nonce validation + checksum generation and verification + two public-key operations + response to the user’s query</td>
</tr>
<tr>
<td>[144]</td>
<td>lookup table query + three hash operations for parameters generation + waiting for the GW-node’s response + response to the user’s query</td>
</tr>
<tr>
<td>[58]</td>
<td>timestamp validation + one hash operation for parameter generation + response to the user’s query</td>
</tr>
<tr>
<td>[116]</td>
<td>timestamp validation + one key derivation function for computing encryption key + one key derivation function operation for computing MAC key + two MAC operations for parameters generation + one encryption for encrypting data + one decryption for retrieving MAC and encryption keys shared between user and sensor node</td>
</tr>
<tr>
<td>[79]</td>
<td>timestamp validation + one hash operation for parameter verification + response to the user’s query</td>
</tr>
<tr>
<td>[73]</td>
<td>timestamp validation + one hash operation for parameter generation + response to the user’s query</td>
</tr>
<tr>
<td>[134]</td>
<td>timestamp validation + one hash operation for parameter verification + one hash operation for parameter generation + response to the user’s query + waiting for the GW-node’s response</td>
</tr>
<tr>
<td>[68]</td>
<td>one hash operation for random nonce validation + one hash operation for session key generation + response to the user’s query</td>
</tr>
<tr>
<td>[23]</td>
<td>timestamp validation + one hash operation for parameter generation + response to the user’s query</td>
</tr>
<tr>
<td>Ours</td>
<td>timestamp validation + two hash operations for parameter generation and symmetric session key generation + one symmetric-key decryption + response to the user’s query + acknowledgment to the BS</td>
</tr>
</tbody>
</table>
not consume battery for user authentication process. Due to efficient hash and symmetric key operations, sensor node’s energy cost in our scheme is comparable with that for other schemes.

4.6 Summary

In this chapter we have proposed a new password-based user authentication scheme for large-scale hierarchical wireless sensor networks. The proposed scheme allows the user to authenticate at both the base station and the cluster heads inside WSN. After successful authentication, both the user and the cluster head from which user wants to access real-time data in the target field, will be able to establish a secret session key between them. Later using this session key, the user can contact the cluster head for real-time data inside WSN. The proposed scheme supports dynamic node addition phase and in that case, there is no need to update stored information in the user’s smart card for accessing real-time data from the newly added cluster heads in the network. Further, our scheme provides better security compared with other related schemes and supports changing of a user’s password locally without contacting the base station. In addition, our scheme is also efficient in terms of communication and computation as compared with those for other related schemes.
Chapter 5

Certificate-Based Access Control in Distributed WSNs

In an access control scheme, a deployed sensor node proves its identity to its neighbor nodes through authentication and also proves that it has the proper right to access the sensor network. After successful authentication, the shared secret keys should be established between a deployed sensor node and its neighbor nodes to protect communications. In a wireless sensor network, we often require deployment of new nodes to extend the lifetime of the network because sensor network may be lost due to power exhaustion problem or malicious nodes. In order to protect malicious nodes from joining the sensor network, access control mechanism becomes a major challenge in the design of sensor network protocols due to resource limitations of sensor nodes.

Existing access control protocols designed for wireless sensor networks require either high communication overheads or they are not scalable due to involvement of the base station during authentication and key establishment processes. In this chapter, we propose a new access control scheme for large-scale distributed wireless sensor networks, which not only identifies the identity of each node but it has also ability to differentiate between old nodes and new nodes. The proposed scheme does not require involvement of the base station during authentication and key establishment processes, and it can be easily implemented as a dynamic access control protocol. In addition, our scheme significantly reduces communication costs in order to authenticate neighbor nodes among each other and establish symmetric keys between neighbor nodes as compared with existing approaches. Further, our
scheme is secure against different attacks and unconditionally secure against node capture attacks.

In this chapter, we first propose a threat model for our scheme. We then provide the notations, which are used in our scheme. We discuss the main motivation behind the development of our novel access control scheme in distributed wireless sensor networks. Finally, we discuss the various phases related to our proposed scheme.

5.1 Threat model

We assume that due to the hostile environments in the deployment field, sensor nodes can be physically captured by an attacker. We also assume that sensor nodes are not equipped with tamper-resistant hardware due to cost constraints and as a result, once a node is captured by an attacker, all the sensitive data as well as cryptographic information including secret keys stored in its memory are revealed to the attacker. The threats associated to security protocols in sensor networks that are implemented in devices located in hostile environments. Most of those threats come from the fact that any potential attacker has physical access to the device implementing the protocols. Hence, it is possible that (s)he has the ability of dumping memory contents, reading firmware etc. Depending on the attacker motivation, hardware invasive or semi-invasive techniques could be applied to recover secret keys. If such keys belong to ordinary devices, then false data could be injected in the WSN. However, we assume that in any case, the base station (BS), which is the central authority (CA) in our case, will not be compromised by an attacker and thus, the attacker does not have any ability to know the private key of the CA. We further assume that an attacker can directly deploy malicious nodes in the deployment field after the initial deployment of nodes.

As in [58], we make use of the Dolev-Yao threat model [61] in which two communicating parties (nodes) communicate over an insecure channel. We adopt the similar threat model for WSNs where the channel is insecure and the end-points (sensor nodes) cannot in general be trustworthy. Finally, we assume that an attacker has only the ability to eavesdrop on all traffic and reply old messages previously delivered.
5.2 Notations

We use the notations for describing our proposed access control scheme as shown in Table 5.1. Random nonce is a one-time random bit-string which is usually used to achieve freshness. The public key of CA is $Q = xG$, where $xG = G + G + \ldots + G$ ($x$ times) is called the elliptic curve scalar multiplication in $E_p(a, b)$. If $nG = O$, where $O$ is the point at infinity or zero point [88], then we call $n$ is the order of the base point $G$ in $E_p(a, b)$. We may use SHA-1 [7] as the hash function $h(\cdot)$.

5.3 Motivation

Our proposed access control scheme is motivated by the following considerations. Compared to RSA, elliptic curve cryptography (ECC) can achieve the same level of
security with smaller key size [135]. In wireless sensor networks, the transmission energy consumption rate is approximately over three orders of magnitude greater than the energy consumption rate for computing [18]. However, currently there exist few transceivers with lower communication for transmission and reception. An example of such a transceiver is CC2420 [5]. The packet size and the number of packets in transmission play a crucial role for the performance while designing an access control protocol in sensor networks. If a node is preloaded with the certificate by the CA (in our scheme, it is the base station), then verifying RSA signature in the certificate takes less time than that for ECC signature verification in the certificate, since the signature will be generated in offline by the CA prior to deployment of sensor nodes in the target field. However, compared to a 1024-bit RSA signature [122], if we use ECC-based signature [82], [102] in certificate, then we require only 320-bit signature when 160-bit ECC is used in the proposed scheme. This motivates us to use ECC instead of RSA in our proposed access control scheme so that we can achieve much more energy and bandwidth savings. Our scheme uses the symmetric key cryptographic techniques along with ECC in order to achieve communication and computational efficiency.

Different access control schemes are proposed in the literature. We have critically analyzed the storage, communication, computational overheads requirement, functionality and security analysis of the existing schemes. Some schemes are vulnerable to different attacks and also some schemes require high storage, communication and computational costs like Zhou et al.’s scheme [151] is not secure against node compromise attack. Further, their scheme assumes that each node can sustain a tolerance time interval before it can be compromised [150], [151] and thus, their scheme may not be convenient for practical implementations [77]. Moreover, their scheme requires high communication and computational overheads due to exchange of so many messages for the entire protocol. Though Huang’s scheme [77] is efficient than [151], it is insecure against the replay attack and an active attack such as new node masquerading attack, and it has the lack of hash chain renewability [85]. Kim-Lee’s scheme [85] is an enhancement of Huang’s scheme [77]. However, their scheme is again vulnerable to active attack. Though the recently proposed Huang’s scheme [78] is secure against different attacks except node capture attack, it requires high storage and computational overheads. In this chapter, we aim to devise an efficient and secure access control scheme. In our scheme, we use a preloaded certificate in
5.4 Our contributions

each sensor node prior to its deployment in the target field. The preloaded certificate in each node contains a version number which is different for each deployment phase for sensor nodes, a unique certificate serial number, the issuer name (CA, that is, base station), bootstrapping time and the node’s identifier. In addition, we use the latest version verified field in each node so that with the help of the bootstrapping time and version present in the certificate, each node will be able to detect a malicious node in the network. After successful authentication with neighbor nodes, each node establishes distinct pairwise symmetric keys with its neighbors for their future secure communication.

5.4 Our contributions

Our scheme has the following attractive properties:

- Our scheme is secure against different attacks. The resilience against node compromise attack of our scheme is much higher than other existing schemes. In addition, our scheme prevents malicious node deployment attack, Sybil attack, node replication attack and wormhole attack as compared to other schemes.
- Our scheme requires less communication, computation, and storage overheads as compared to other existing schemes.
- Our scheme does not require any involvement of the base station during the authentication and key establishment phase as well as dynamic node addition phase.
- Higher security along with lower communication, computation and storage overheads make our scheme much suitable for practical applications in WSN.

5.5 The proposed certificate-based access control scheme

In this section, we describe the various phases related to our proposed scheme. In our scheme, a sink node or base station acts as a certificate authority [85], [151] and
preloads that certificate in each sensor node’s memory prior to its deployment in the target field. The preloaded certificate in each node contains a version number, which is different for each deployment phase for sensor nodes, a unique certificate serial number, the issuer name (CA, that is, base station), bootstrapping time and the node’s identifier. In addition, we use the latest version verified field in each sensor node so that with the help of the bootstrapping time and version present in the certificate, each sensor node will be able to detect whether a deployed node is a malicious node in the network. As a result, the base station is treated as a central certificate authority [85], [151]. The various phases of our scheme are described in the following subsections.

5.5.1 Pre-deployment phase

This phase is performed by the CA (the base station in our scheme) in offline before deployment of sensor nodes in a particular deployment field (target field). The pre-deployment phase consists of the following steps:

Step 1: Prior to deployment of sensor nodes in the target field, the CA chooses a set of network parameters which includes (i) a finite field $GF(p)$, where $p$ is a large odd prime of at least 160-bits; (ii) an elliptic curve $E_p(a, b)$, which is the set of all points of $y^2 = x^3 + ax + b \pmod{p}$ such that $a, b \in \mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$ are constants with $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$; (iii) a base point $G$ in $E_p(a, b)$ whose order is $n$, where $n$ is at least 160-bit number such that $n > 4\sqrt{p}$ as in [151] and $n$ is also prime; (iv) the CA’s private key $x \in \mathbb{Z}_n^*$, where $\mathbb{Z}_n^* = \{1, 2, \ldots, n-1\}$; (v) the CA’s public key $Q = xG$.

Step 2: Once the set of network parameters are selected, the CA preloads a set of node parameters for each sensor node $u_i$ prior to its deployment in offline. This set contains (i) a unique node identifier $id_{u_i}$ of the node $u_i$; (ii) the elliptic curve $E_p(a, b)$; (iii) the base point $G$; (iv) the certificate $Cert_{u_i}$ for node $u_i$ shown in Table 5.2; (v) the CA’s public key $Q$; (vi) a secure one-way hash function $h(\cdot)$; (vii) a variable called latest_version_verified.

A preloaded certificate in each sensor node $u$ prior to its deployment is used to prove its own identity to its neighbor nodes. The purpose of the preloaded certificate in each sensor’s memory is that when a new node is deployed in the sensor network,
5.5 The proposed certificate-based access control scheme

Table 5.2: Certificate of a node $u$.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version ($V$)</td>
<td>Integer</td>
</tr>
<tr>
<td>Certificate serial number ($SN$)</td>
<td>Integer</td>
</tr>
<tr>
<td>Issuer name ($CA$)</td>
<td>String</td>
</tr>
<tr>
<td>Bootstrapping time ($T_u$)</td>
<td>Time</td>
</tr>
<tr>
<td>Node identifier ($id_u$)</td>
<td>String</td>
</tr>
<tr>
<td>Signature on all above fields ($sig$)</td>
<td>ECDSA signature</td>
</tr>
</tbody>
</table>

its neighbor nodes can verify the certificate in order to check whether the new node is a legitimate or not. The structure of a preloaded certificate $Cert_u$ in a sensor node $u$ is shown in Table 5.2, which has similar structure as in [151]. Note that this proposed certificate is completely different from the traditional certificate (for example, X.509 [130]), which is much suitable for the resource-constrained sensor nodes to keep the minimal information needed for neighbor sensor node authentication by a sensor node. The version field ($V$) is different for each certificate preloaded in sensor nodes. The version $V$ is initialized as follows:

$$V = \begin{cases} 
1, & \text{if } u \text{ is deployed during initial deployment phase} \\
 i, & \text{if } u \text{ is deployed during } i\text{-th dynamic nodes addition phase.} 
\end{cases}$$

Each certificate will have a unique serial number ($SN$). In our scheme, the issuer ($CA$) in each certificate is basically the base station. Each certificate will contain a bootstrapping time, $T_u$ so that the node $u$ bootstraps itself to join the sensor network. The certificate will contain the unique node identifier $id_u$ for a sensor node $u$. Finally, the signature on all above fields is computed using the elliptic curve digital signature algorithm (ECDSA) [82], [102] with the help of the private key $x$ of the CA. Since the private key $x$ of the CA is only known to the CA and no one (including the sensor nodes and attackers) can derive $x$ from the public key $Q = xG$ of the CA due to difficulty in solving the elliptic curve discrete logarithm problem (ECDLP), the CA can only generate the valid certificates for deployed sensor nodes.

The purpose of using the variable, $\text{latest}_\text{version}_\text{verified}$ is that when a new node is deployed in the network, the old existing nodes will expect that their
latest\_version\_verified should be at least the version of the certificate of the new deployed node. This variable of a node $u$ is initialized by the CA prior to deployment as $\text{latest}\_\text{version}\_\text{verified} = V$.

After deployment, once the node $u$ authenticates with its neighbor nodes and establishes the pairwise symmetric secret keys with neighbors, the variable $\text{latest}\_\text{version}\_\text{verified}$ will be updated by $u$ as discussed in Section 5.5.2.

### 5.5.2 Authentication and key establishment phase

This phase is executed by each deployed sensor node in the network. Initially, a large number of sensor nodes are deployed in the target field. We can deploy some new sensor nodes in the existing sensor network when some nodes may exhaust their power or they can be compromised by an attacker.

The main purpose of this phase is that each sensor node will authenticate its neighbor nodes in its communication range and also establish secret pairwise symmetric keys with its neighbors after successful mutual authentication. The first task of a deployed sensor node is to locate its neighbor nodes in its communication range. In order to know neighbor nodes, each node $u$ broadcasts a HELLO message containing its own identifier $\text{id}_u$. If the node $u$ receives $d$ HELLO messages, then it prepares a list of neighbors as $\text{NL}_u = \{v_1, v_2, \ldots, v_d\}$, where $\text{id}_{v_1}, \text{id}_{v_2}, \ldots, \text{id}_{v_d}$ are the corresponding identifiers of the neighbor nodes $v_1, v_2, \ldots, v_d$, respectively. The node $u$ will authenticate its $d$ neighbors in $\text{NL}_u$ and if the authentication be successful, it will establish secret pairwise symmetric keys with them. A typical value of number of neighbors of a sensor node in WSN is $d = 40$ [153].

Let $u$ and $v$ be two neighbor nodes. The authentication and key establishment procedure between $u$ and $v$ involves the following steps:

**Step 1:** Node $u$ generates a random nonce $RN_u$ and a random secret number $r_u(< n)$. Note that $n$ is the order of the base point $G$ in the elliptic curve $E_p(a, b)$. $r_u$ is $u$’s private key. It computes the public key $Q_u = r_uG$. $u$ then sends the following message to its neighbor node $v$:

$$u \rightarrow v : (\text{id}_u||RN_u||Q_u||\text{Cert}_u).$$

**Step 2:** After receiving the message from $u$, $v$ verifies the certificate $\text{Cert}_u$ of $u$ by means of verifying the signature present in that certificate using the ECDSA
signature verification algorithm by the CA’s public key $Q$. If the signature verification is successful, then $v$ further verifies $id_u$ with the received identity of the certificate of $u$. If they match, $v$ assumes that node $u$’s identity is valid. We have then the following three cases for the node $v$.

Case 1: $T_v = T_u$

In this case, the bootstrapping time $T_v$ of node $v$ present in its certificate is equal to the bootstrapping time $T_u$ of node $u$ present in the certificate. If $Cert_v.V = Cert_u.V$ and the value of latest \_version\_verified for node $v$ is also equal to $Cert_u.V$, then node $v$ ensures that node $u$ is deployed during the same deployment phase. Here $Cert_u.V$ represents the version in the certificate of node $u$. Thus, nodes $u$ and $v$ are both new nodes. Node $v$ accepts node $u$ as a legitimate node in the network.

Case 2: $T_v > T_u$

Node $v$ verifies whether $Cert_v.V > Cert_u.V$ and the value of latest \_version\_verified for node $v$ is greater than or equal to the version available in $Cert_u$. If both conditions are satisfied, then $u$ is accepted as a legitimate node by the node $v$. In such case, node $u$ is considered as old node and $v$ is considered as new deployed node in the network.

Case 3: $T_v < T_u$

Node $v$ verifies whether $Cert_v.V < Cert_u.V$ and the value of latest \_version\_verified for node $v$ is less than or equal to the version available in $Cert_u$. If both conditions are satisfied, then $u$ is also accepted as a legitimate node by the node $v$. Here, node $v$ is considered as old node and node $u$ is considered as new deployed node in the network.

Note that the purpose of comparing $T_u$ with $T_v$ is as follows. When $T_v = T_u$, both nodes $u$ and $v$ are treated as new nodes and $v$ accepts $u$ as a legitimate node in the network. When $T_v > T_u$, node $u$ is considered as old node and $v$ is considered as new deployed node in the network. On the other hand, when $T_v < T_u$, node $v$ is considered as old node and node $u$ is considered as new deployed node in the network. Thus, it is clear that by this mechanism, we are able to detect a malicious deployed sensor node in the network.

Node $v$ now generates a random nonce $RN_v$ and a random secret number $r_v(< n)$, which is considered as the secret key of node $v$. $v$ computes the public
key $Q_v = r_v G$. After that $v$ computes $r_v Q_u = (K_{x_{vu}}, K_{y_{vu}})$ and computes the
secret symmetric shared key $K_{v,u}$ with node $u$ as

$$K_{v,u} = h(id_v || id_u || RN_u || RN_v || T_u || T_v || K_{x_{vu}} || K_{y_{vu}}). \quad (5.1)$$

For the challenge-response protocol, $v$ can create a puzzle message, say $PM$ and then computes the encrypted puzzle using its computed key $K_{v,u}$ as $E_{K_{v,u}}(PM)$ and the hash value $h(K_{v,u} || PM || RN_u)$. Finally, $v$ sends the following message to node $u$:

$$v \rightarrow u: \langle id_v || RN_u || RN_v || Q_v || Cert_v || E_{K_{v,u}}(PM) || h(K_{v,u} || PM || RN_u) \rangle.$$

Step 3: After receiving the message from node $v$, node $u$ proceeds as follows. $u$ first verifies the certificate $Cert_v$ of $v$ received in the message by means of verifying the signature containing in $Cert_v$ using the ECDSA signature verification algorithm by the CA’s public key $Q$. If the signature verification is successful, $u$ further verifies the identity $id_v$ of node $v$ with the identity present in $Cert_v$ and the received random nonce $RN_u$ in the message with its own previously generated random nonce for authentication with node $v$. If these verifications are successful, then $u$ assumes that node $v$’s identity is valid.

Similar to Step 2, we have also the following three cases for the node $u$.

Case 1: $T_u = T_v$

If $Cert_u.V = Cert_v.V$ and the value of latest version verified for node $u$ is equal to $Cert_v.V$, then node $u$ also ensures that node $v$ is deployed during the same deployment phase. In this case, nodes $u$ and $v$ are both new nodes. Node $u$ then accepts node $v$ as a legitimate node in the network.

Case 2: $T_u > T_v$

In this case, $u$ verifies whether $Cert_u.V > Cert_v.V$ and the value of latest version verified for node $u$ is greater than or equal to the version available in $Cert_v$. If both conditions are satisfied, then $v$ is accepted as a legitimate node by the node $u$, and as a result, node $v$ is considered as old node and node $u$ is considered as new deployed node in the network.
Case 3: \( T_u < T_v \)

Node \( v \) verifies whether \( \text{Cert}_u.V < \text{Cert}_v.V \) and the value of \textit{latest\_version\_verified} for node \( u \) is less than or equal to the version available in \( \text{Cert}_v \).

Now, if both these conditions are satisfied, then \( v \) is also accepted as a legitimate node by the node \( u \). Hence, node \( u \) is considered as old node and node \( v \) is considered as new deployed node in the network.

Once the node \( u \) considers the node \( v \) as a legitimate node, \( u \) computes \( r_u Q_v = (K_{x_{uv}}, K_{y_{uv}}) \). \( u \) then computes the symmetric secret key \( K_{u,v} \) shared with \( v \) as

\[
K_{u,v} = h(id_u || id_v || RN_u || RN_v || T_u || T_v || K_{x_{uv}} || K_{y_{uv}}).
\]

(5.2)

In order to solve the puzzle, \( u \) first decrypts the encrypted puzzle \( E_{K_{v,u}}(PM) \) using its own computed key \( K_{u,v} \) and retrieves the puzzle as \( PM' = D_{K_{u,v}}(E_{K_{v,u}}(PM)) \) and computes the hash value using the retrieved puzzle \( PM' \), its own computed key \( K_{u,v} \) and its own random nonce \( RN_u \) as \( h(K_{u,v} || PM' || RN_u) \).

If this computed hash value matches with the incoming hash value \( h(K_{v,u} || PM || RN_u) \) received in the message, then \( u \) ensures that it has solved the puzzle successfully. \( u \) then executes Step 4.

Step 4: To make sure that node \( u \) has successfully computed the same secret key shared with node \( v \) and solved the puzzle sent by node \( v \), it computes the hash value as \( h(K_{v,u} || PM' || RN_v) \). Node \( u \) then sends the following message to \( v \) as a response of the previous challenge sent by node \( v \):

\[
u \rightarrow v : \langle id_u || id_v || RN_v || h(K_{u,v} || PM' || RN_v) \rangle.
\]

Step 5: After receiving the message from node \( u \), node \( v \) first verifies whether the random nonce received in the message matches with its own random nonce \( RN_v \). If so, node \( v \) computes the hash value \( h(K_{v,u} || PM || RN_v) \) using its own computed key \( K_{v,u} \), previously created puzzle \( PM \) and random nonce \( RN_v \).

If this hash value matches with the hash value received in the message, then \( v \) also ensures that node \( u \) has the correct same secret key shared between them and it has solved the puzzle successfully.

Node \( u \) stores the secret key \( K_{u,v} \) in its memory for future secret communication with node \( v \). Similarly, node \( v \) also stores the secret key \( K_{v,u} \) in its memory for future secret communication with node \( u \).
This phase is summarized in Table 5.3. We note that our scheme provides mutual authentication between any two neighbor nodes in the network.

Table 5.3: Authentication and key establishment phase of our proposed scheme.

<table>
<thead>
<tr>
<th>Node u</th>
<th>Node v</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Generates $RN_u$, $r_u$. Computes $Q_u = r_uG$. (\langle id_u</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. If above verifications hold, generates $RN_v$, $r_v$, computes $r_vQ_u = (K_{x,v}, K_{y,u}), K_{v,u} = h(id_u</td>
</tr>
<tr>
<td></td>
<td>4. Verifies $Cert_v$. Verifies version, bootstrapping time present in $Cert_v$, $latest_version_verified$ of $u$.</td>
</tr>
<tr>
<td></td>
<td>5. If above verifications hold, computes $r_uQ_v = (K_{x,v}, K_{y,u}), K_{u,v} = h(id_u</td>
</tr>
<tr>
<td></td>
<td>6. Verifies received random nonce $RN_v$ with its own random nonce. If so, computes $h(K_{v,u}</td>
</tr>
</tbody>
</table>

In order to thwart against node capture attacks, node $u$ deletes $RN_u$, $r_u$, $Q_u$ from its memory and node $v$ also deletes $RN_v$, $r_v$, $Q_v$ immediately after their key establishment. After successful authentication and key establishment with neighbor nodes, each sensor node $u$ will first set its variable $latest\_version\_verified$ by the latest version of a certificate of its neighbors nodes and then increment it. For
5.5 The proposed certificate-based access control scheme

example, \( u \) updates its variable `latest_version_verified` by 2 after authenticating and establishing secret keys with its neighbors during the initial deployment phase. This means that \( u \) is now ready to authenticate and establish secret keys with only a new deployed node during \( i \)-th dynamic nodes addition phase (\( i \geq 2 \)) whose certificate’s version is greater than or equal to 2.

**Remark 5.1:** In our proposed scheme, the sensor node \( v \) computes \((K_{xvu}, K_{yvu})\) using \( r_v \) and \( Q_u \) and this computed value \((K_{xvu}, K_{yvu})\) is used for generating the secret symmetric shared key \( K_{v,u} \) with its neighbor node \( u \) as \( K_{v,u} = h(id_u || id_v || RN_u || RN_v || T_u || T_v || K_{xvu} || K_{yvu}) \), where the other parameters \( id_u, id_v, RN_u, RN_v, T_u, T_v \) are taken from both the preloaded certificates of \( u \) and \( v \) which can not be changed as ECDSA signature generated by the base station (the CA) is used in the certificate. Moreover, for the challenge-response protocol, the node \( v \) creates a puzzle message, say \( PM \) and then computes the encrypted puzzle using its computed key \( K_{v,u} \) as \( E_{K_{v,u}}(PM) \) and the hash value \( h(K_{v,u} || PM || RN_u) \). To verify the puzzle message, node \( u \) has to compute \( K_{xvu}, K_{yvu} \) and \( K_{u,v} \) using other parameters taken from both the preloaded certificates of \( u \) and \( v \) and then decrypt the encrypted puzzle \( E_{K_{v,u}}(PM) \) using its own computed key \( K_{u,v} \). After that \( u \) retrieves the puzzle as \( PM' = D_{K_{u,v}}(E_{K_{v,u}}(PM)) \) and computes the hash value using the retrieved puzzle \( PM' \), its own computed key \( K_{u,v} \) and its own previously generated random nonce \( RN_u \) as \( h(K_{u,v} || PM' || RN_u) \). If this computed hash value matches with the incoming hash value \( h(K_{v,u} || PM || RN_u) \) received in the message, then only \( u \) ensures that it has solved the puzzle successfully. To make sure that node \( u \) has successfully computed the same secret key shared with node \( v \) and solved the puzzle sent by node \( v \), it computes the hash value as \( h(K_{u,v} || PM' || RN_u) \) and sends it to \( v \) as a response of the previous challenge sent by node \( v \). After receiving the message from node \( u \), node \( v \) first verifies whether the random nonce received in the message matches with its own random nonce \( RN_v \). If so, node \( v \) computes the hash value \( h(K_{v,u} || PM || RN_u) \) using its own computed key \( K_{v,u} \), previously created puzzle \( PM \) and random nonce \( RN_u \). If this hash value matches with the hash value received in the message, then \( v \) also ensures that node \( u \) has the correct same secret key shared between them and it has solved the puzzle successfully. Hence, it is clear that \( Q_u \) and \( Q_v \) can not be modified by an adversary during the transmission between nodes \( u \) and \( v \).
5.5.3 Dynamic nodes addition phase

Deployment of new nodes in sensor networks is inevitable due to the loss of sensor nodes because after several weeks or months of operation some sensor nodes in the network may exhaust their power. Even some nodes may be compromised and we need to deploy new nodes in the network. Assume that one or more new nodes to be deployed in a dynamic nodes addition phase.

Let a new sensor node \( u \) be deployed in the \( i \)-th dynamic nodes addition phase. Prior to its deployment in the target field, using the pre-deployment phase discussed in Section 5.5.1, the CA will preload a set of node parameters in offline. This set contains (i) a unique node identifier \( id_u \) of the node \( u \); (ii) the elliptic curve \( E_p(a,b) \); (iii) the base point \( G \); (iv) the certificate \( Cert_u \) for node \( u \) shown in Table 5.2; (v) the CA’s public key \( Q \); (vi) a secure one-way hash function \( h(\cdot) \); (vii) a variable called \( latest\_version\_verified \) initialized to \( i \), for the \( i \)-th dynamic nodes addition phase.

After deployment, \( u \) authenticates and establishes pairwise symmetric secret keys with its neighbor nodes using the authentication and key establishment phase as described in Section 5.5.2. Thus, in our scheme, dynamic nodes addition phase is very simple and efficient, and it does not require any involvement of the base station after deployment.

5.6 Analysis of the proposed scheme

In this section, we perform the functionality analysis and security analysis including the formal security analysis of our proposed access control scheme.

5.6.1 Functionality analysis

Correctness proof

We show that any two neighbor nodes in the network can establish correctly a symmetric secret key shared between them.

This correctness proof is given in Theorem 5.1.

Theorem 5.1. In our access control scheme, any two neighbor sensor nodes always establish correctly the same symmetric secret key shared between them after successful
mutual authentication between them.

Proof. Let $u$ and $v$ be two neighbor sensor nodes. From Section 5.5.2, it is clear that node $u$ establishes a symmetric secret key with node $v$ as $K_{u,v} = h(id_u||id_v||RN_u||RN_v||T_u||T_v||K_{x_{uv}}||K_{y_{uv}})$, where $r_u Q_v = (K_{x_{uv}}, K_{y_{uv}})$. On the other hand, node $v$ also establishes a symmetric secret key with node $u$ as $K_{v,u} = h(id_u||id_v||RN_u||RN_v||T_u||T_v||K_{x_{vu}}||K_{y_{vu}})$, where $r_v Q_u = (K_{x_{vu}}, K_{y_{vu}})$. In order to prove $K_{u,v} = K_{v,u}$, we have to show that $K_{x_{uv}} = K_{x_{vu}}$ and $K_{y_{uv}} = K_{y_{vu}}$, that is, it suffices to show that $r_u Q_v = r_v Q_u$.

Now we have,

$$r_u Q_v = r_u (r_v G) = r_v (r_u G) = r_v Q_u.$$

Hence, $u$ and $v$ always establish correctly the same symmetric secret key shared between them. \qed

Secure connectivity

We measure the secure connectivity of our access control scheme by the probability that any two neighbor sensor nodes can establish a secret pairwise key (secure link) between them. It is noted that any two neighbor sensor nodes can establish a symmetric secret key shared between them, if they authenticate each other successfully. As a result, the secure connectivity probability becomes 1.0. Hence, our scheme provides 100% secure connectivity in the network.

Storage overhead

Prior to deployment in the pre-deployment phase, each sensor node $u$ requires the storage space due to storing the node parameters: a unique node identifier $id_u$ of the node $u$; the elliptic curve $E_p(a, b)$; the base point $G$; the certificate $Cert_u$ for node $u$ shown in Table 5.2; the CA’s public key $Q$; and the variable $latest_version_verified$. Using Table 5.4, the storage overhead for each node $u$ of our scheme becomes 1560 bits.
Table 5.4: Bit size of the parameters used for our proposed access control scheme.

<table>
<thead>
<tr>
<th>Type</th>
<th>Bit size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version (V)</td>
<td>8</td>
</tr>
<tr>
<td>Serial number (SN)</td>
<td>32</td>
</tr>
<tr>
<td>Issuer name (CA)</td>
<td>8</td>
</tr>
<tr>
<td>Bootstrapping time (Tu)</td>
<td>32</td>
</tr>
<tr>
<td>Node identifier (id_u)</td>
<td>16</td>
</tr>
<tr>
<td>Signature (ECDSA)</td>
<td>320</td>
</tr>
<tr>
<td>Random nonce (RN_u)</td>
<td>32</td>
</tr>
<tr>
<td>Hash digest</td>
<td>160</td>
</tr>
<tr>
<td>Latest version verified</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 5.5: Message size in bits and number of packets needed to transmit during authentication and key establishment phase of our proposed access control scheme.

<table>
<thead>
<tr>
<th>Message</th>
<th>Number of bits required</th>
<th>Number of packets required</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨id_u</td>
<td></td>
<td>RN_u</td>
</tr>
<tr>
<td>⟨id_v</td>
<td></td>
<td>RN_u</td>
</tr>
<tr>
<td></td>
<td>1104</td>
<td>2</td>
</tr>
<tr>
<td>⟨id_u</td>
<td></td>
<td>id_v</td>
</tr>
</tbody>
</table>

Communication overhead

It is clear from the authentication and key establishment phase described in Section 5.5.2 that any two neighbor nodes need to exchange only three messages in order to authenticate with each other and establish a symmetric secret key between them. We use Table 5.4 for different parameters used for our access control scheme. We have calculated the size of each message in bits and then the number of packets required to transmit per message. For calculating the number of packets required for a message, we have considered the packet size of 128 bytes, that is, 1024 bits for CC2420. Note that the message ⟨id_v||RN_u||RN_v||Q_v||Cert_v||EK_v,u(PM) ||h(K_v,u||PM||RN_u))⟩ re-
5.6 Analysis of the proposed scheme

quires 1104 bits, which are equivalent to 1024 + 80 bits = 1 + 1 = 2 packets. From Table 5.5, we see that our scheme requires only 4 packet transmissions during the the authentication and key establishment phase.

Computational overhead

We consider the computational complexity for a sensor node $u$ for authentication and key establishment of a symmetric secret key with a neighbor node $v$. Let $T_{ecm}$, $T_h$, $T_i$, $T_{enc}$, and $T_{dec}$ denote the time for performing an elliptic curve scalar multiplication, a one-way hash function $h(\cdot)$, a modular inverse, a symmetric key encryption, and a symmetric key decryption, respectively. Node $u$ needs the computational complexity due to the following: computation of the public key $Q_u$, which requires $T_{ecm}$; verification of the certificate of $v$, which requires $2T_{ecm} + T_i + T_h$; computation of symmetric key, which requires $T_{ecm} + T_h$; computation for the challenge-response protocol, which requires $T_{enc}/T_{dec} + 2T_h$. Summing up all these complexity terms together, $u$ requires $4T_{ecm} + T_i + T_{enc}/T_{dec} + 4T_h$.

5.6.2 Security analysis

We show that our scheme can effectively defend various attacks, which are discussed in the following subsections.

Withstand external devices to eavesdrop or inject data

In our scheme, when a new sensor node passes the successful authentication procedure, it has already established the shared symmetric secret keys with its neighbors. These secret keys can be used to secure communications among its neighbor sensor nodes. As a result, external devices are prevented from eavesdropping or injecting false reports into the sensor network.

Resilience against node capture attacks

In the following, we have evaluated the ability of our access control scheme to tolerate compromised sensor nodes in the network. Let $P_e(c)$ denote the probability that an adversary compromises a fraction of total secure communications by capturing $c$ number of sensor nodes in the network. If $P_e(c) = 0$, we call an access control scheme as unconditionally secure against node capture attack.
Our scheme ensures that any two neighbor sensor nodes can establish a symmetric secret key between them for their future secure communications. Moreover, since the pairwise key between any two neighbor nodes is generated using random private keys by the nodes, the established secret keys among sensor nodes are different throughout the network. If we have more neighbor nodes of each node, the number of secure links in the network also increases. Now, if a sensor node is captured or compromised, then the attacker is only able to compromise secret keys of its neighbor nodes. The effect of a compromised node does not lead to compromise secure communications among non-compromised nodes in the network. In other words, even some nodes are compromised, the security of the entire sensor network is not compromised, that is, non-compromised nodes still communicate among each other with 100% security. Hence, our scheme is unconditionally secure against node capture attack.

**Resilience against new node deployment attacks**

Through the formal security analysis, we show that our scheme has the ability to tolerate various known attacks. We show that our proposed scheme prevents malicious node deployment attack, Sybil attack, node replication attack and wormhole attack by an adversary.

We follow the similar proof as in [20], [21], [42], [53], [112], [117], [118] for the formal security analysis of our proposed scheme. We have used the method of contradiction proof [25] for our formal security analysis. Note that one can also prove the formal security in the standard model. However, in this thesis, we have performed the formal security analysis under the generic group model of cryptography. For this purpose, we assume that the following two oracles exist for an adversary $A$:

- **RevealKey**: This unconditionally outputs the private key $x$ of the CA using the public elliptic curve parameters and the CA’s public key $Q = xG$.

- **CreateCertificate**: This query is allowed at any time during the attacker’s execution. This oracle computes a valid certificate of a new deployed malicious node on behalf of the CA.

**Theorem 5.2.** Under the elliptic curve discrete logarithm problem (ECDLP) assumption, our proposed scheme prevents malicious node deployment attack, Sybil attack, node replication attack and wormhole attack by an adversary.
5.6 Analysis of the proposed scheme

Proof. We use the method of contradiction proof [25] for proving this theorem. Assume that an adversary can determine the private key $x$ of the CA from the public elliptic curve parameters and the CA’s public key $Q = xG$. Thus, the adversary has the ability to generate a valid certificate with proper bootstrapping time, version of the certificate, certificate serial number and node identifier for a new deployed malicious node in the network on behalf of the CA. For this purpose, an adversary $A$ can run the following experimental algorithm, say, $\text{Experiment}^{\text{ECDLP}}_{\text{ACP}, A}$ for our proposed access control protocol $ACP$, which is provided in Algorithm 4, and uses both random oracles RevealKey and CreateCertificate.

Algorithm 4 $\text{Experiment}^{\text{ECDLP}}_{\text{ACP}, A}$

1: Call RevealKey oracle. Let $x' \leftarrow \text{RevealKey}(E_p(a, b), n, G, Q)$ be the output, which be the private key of the CA.
2: Call CreateCertificate oracle.
   Let $\text{Cert}_w \leftarrow \text{CreateCertificate}(x', E_p(a, b), n, G)$ be the output, which be a certificate generated for a new malicious deployed node $w$ by the adversary $A$ using the private key computed by the RevealKey oracle.
3: if (CreateCertificate oracle generates correctly a valid certificate for the node $w$) then
   4: return 1 (Success)
5: else
6: return 0 (Failure)
7: end if

We define $\text{Succ}_{\text{ACP}, A}^{\text{ECDLP}} = 2 Pr[\text{Experiment}^{\text{ECDLP}}_{\text{ACP}, A} = 1] - 1$. Then the advantage function is defined by $\text{Adv}_{\text{ACP}}^{\text{ECDLP}}(t, q_R, q_C) = \max_A \{ \text{Succ}_{\text{ACP}, A}^{\text{ECDLP}} \}$, where the maximum is taken over all $A$ with execution time $t$, $q_R$ is the number of queries to the RevealKey oracle and $q_C$ the number of queries to the CreateCertificate oracle.

We say that our proposed access control scheme is secure against malicious node deployment attack, Sybil attack, node replication attack and wormhole attack, if $\text{Adv}_{\text{ACP}}^{\text{ECDLP}}(t, q_R, q_C) \leq \epsilon$, for any sufficiently small $\epsilon > 0$.

Note that according to the experiment given in Algorithm 4, the adversary $A$ can compute correctly the private key $x$ of the CA. Once the private key $x$ of the CA is known to the adversary $A$, he/she can easily generate a valid certificate for a new malicious deployed node. Now, the new deployed node can authenticate
and establish successfully the secret pairwise keys with its neighbor nodes in the network. Thus, by including the proper bootstrapping time and version of the certificate a new malicious node can easily join to the network. However, this is a contradiction because of the ECDLP assumption defined in Definition 2.3. Computing the private key $x$ of the CA from the public elliptic curve parameters and the public key $Q = xG$ of the CA, it is computationally infeasible problem. As a result, $Adv_{\text{ECDLP}}^{\text{ACP}}(t, q_R, q_C) \leq \epsilon$, for any sufficiently small $\epsilon > 0$. Because the adversary does not know the private key of the CA, he/she cannot falsify certificates for malicious nodes. Since our scheme uses the proper bootstrapping time and version of the certificate, a new malicious deployed node is not allowed to join the sensor network. New malicious deployed nodes are prevented from falsifying the latest bootstrapping time and version of the certificate because they do not match with their certificates. Hence, our access control scheme defends against malicious node deployment attack, Sybil attack, node replication attack and wormhole attack by an adversary.

**Remark 5.2:** Let $u$ and $v$ be two nodes. If $T_u < T_v$, then $u$ is considered as old node and $v$ as new. In our scheme, node $u$ rejects new node $v$, if $\text{Cert}_v.V \not\succ \text{Cert}_u.V$ and the value of $\text{latest\_version\_verified}$ for node $v$ is not greater than or equal to the version available in $u$’s certificate $\text{Cert}_u$. Moreover, if a non-compromised node, say $u$ wants to communicate with compromised node, say $v$ or vice-versa, then in our scheme both nodes $u$ and $v$ must authenticate to each other and also establish a symmetric key between them so that they can use that key for future secret communication. However, $v$ could not be able to pass successfully authentication procedure with $u$ due to timestamp, version and the value of $\text{latest\_version\_verified}$ verifications, because $v$ will not have a valid certificate with proper timestamp and version in the certificate. Hence, our scheme has ability to defend such cases.

### 5.7 Formal security verification of our scheme using AVISPA tool

In this section, we show that our proposed scheme is secure against an adversary using the widely-accepted and used AVISPA (Automated Validation of Internet Security Protocols and Applications) tool [68].
5.7 Formal security verification of our scheme using AVISPA tool

Our protocol is described using a high level language, called the HLPSL language, which is a role-oriented language. Each principal is implemented in transitional roles in which the transitions of a principal takes place during the protocol run as specified. The protocol session is a parallel composition of these transitional roles. The intruder is modeled using the Dolev-Yao threat model [61] (according to our threat model in Section 5.1) with the possibility for the intruder to assume a legitimate role in a protocol run. The role system defines the number of sessions, the number of principals and the roles. In this analysis, the role system consists of two sessions in which the principals, say Alice and Bob take alternatively the roles for initiator and responder, respectively.

5.7.1 Specifying the protocol

We have implemented our scheme under AVISPA model checkers using SPAN (Security Protocol Animator for AVISPA). In our implementation, we have two basic roles: alice and bob, which represent the initiator, sensor node $u$ and the responder, sensor node $v$ respectively. We have further specified the session and environment in our implementation.

Figure 5.1 shows the specification in HLPSL language for the initiator, sensor node $u$. Node $u$ first sends the message $\langle id_u||RN_u||Q_u||Cert_u \rangle$ to the responder, node $v$ with the help of the $Snd()$ operation. Here the type declaration channel $(dy)$ indicates that the channel is for the Dolev-Yao threat model. Node $u$ receives the message $\langle id_v||RN_u ||RN_v||Q_v||Cert_v||E_{K_{v,u}}(PM)||h(K_{v,u}||PM||RN_u) \rangle$ from $v$ with the help of the $Rcv()$ operation. The intruder has the ability to intercept, analyze, and/or modify messages transmitted over the insecure channel. After successful verification node $u$ accepts node $v$ as a legitimate node and then computes $h(K_{u,v}||PM'||RN_v)$ using the symmetric secret key $K_{u,v}$ and then sends the message $\langle id_u||id_v||RN_v ||h(K_{u,v}||PM'||RN_v) \rangle$ to node $v$.

In Figure 5.2, the specification in HLPSL language for the responder (node $v$) is given. After receiving the message from $u$, node $v$ first verifies certificate $Cert_u$ and also verifies version no, bootstrapping time present in $Cert_u$ and latest version verified. After successful verification node $u$ computes the symmetric secret key $K_{v,u}$ and also generates a puzzle $PM$. Then it sends the message $\langle id_v||RN_u||RN_v||Q_v||Cert_v||E_{K_{v,u}}(PM)||h(K_{v,u}||PM||RN_u) \rangle$ to $u$ with the help of the $Snd()$ operation. It then waits for an acknowledgment $\langle id_u||id_v||RN_v ||h(K_{u,v}|| PM'||RN_v) \rangle$
role alice (U,V : agent,
   Kca : public_key,
   % inv(Kca) = private key of CA
   % H is hash function
   H : function,
   F : function,
   IDu, Vru, Cu, Sru, Tu: text,
   Snd, Rcv: channel(dy))
played_by U
def=
   local State : nat,
   IDv, Vrv, Cv, Srv, Tv, Qu, P, Nu, G,
   Ru, Rv : text
   const alice_bob_nv, bob_alice_nu, subs1, subs2 : protocol_id
   init State := 0

transition
1. State = 0 \land Rcv(start) =>
   State' := 1 \land Nu' := new()
      \land Ru' := new()
      \land Qu' := F(Ru'.G)
      \land secret([Ru'], subs1, U)
      \land Snd(U, Nu'.Qu', Vru.Sru.Cu.Tu.IDu.
         [Vru.Sru.Cu.Tu.IDu]_inv(Kca))
      \land witness(U, V, bob_alice_nu, Nu')
2. State = 1 \land Rcv(V.Nv'.Nu'.Qv'.Vrv.Srv.Cv.Tv.IDv
      \land request(V, U, alice_bob_nv, Nu')
   State' := 3 \land Snd(U, V.Nv'.H(H(IDv.IDu.Nv'.Nu'.Tv.Tv.Qv').P'.Nu')), Nu'.P'.Nv'))
   \land request(V, U, alice_bob_nv, Nu')
end role

Figure 5.1: Role specification for the initiator (sensor node \( u \)) for our protocol.

from \( u \). In the roles, mutual authentication between \( u \) and \( v \) are performed.

In HLPSL specification, witness(A,B,id,E) declares for a (weak) authentication property of \( A \) by \( B \) on \( E \), declares that agent \( A \) is witness for the information \( E \); this goal will be identified by the constant \( id \) in the goal section. request(B,A,id,E) means for a strong authentication property of \( A \) by \( B \) on \( E \), declares that agent \( B \) requests a check of the value \( E \); this goal will be identified by the constant \( id \) in the goal section. The intruder is always denoted by \( i \).

We have given the specifications in HLPSL for the role of session, goal and environment in Figures 5.3 and 5.4. In the session segment, all the basic roles: alice, bs and bob are instanced with concrete arguments. The top-level role (environment) defines in the specification of HLPSL, which contains the global constants and a
5.7 Formal security verification of our scheme using AVISPA tool

Figure 5.2: Role specification for the responder (sensor node \(v\)) for our protocol.

composition of one or more sessions, where the intruder may play some roles as legitimate users. The intruder also participates in the execution of protocol as a concrete session. The current version of HLPSL supports the standard authentication and secrecy goals. In our implementation, the following five secrecy goals and two authentications are verified:

- \texttt{secrecy\_of\_subs1}: It represents that \(R_u\) is kept secret to the node \(u\).
- \texttt{secrecy\_of\_subs2}: It represents that \(R_v\) is kept secret to the node \(v\).
- \texttt{authentication\_on\_bob\_alice\_nu}: \(u\) generates a random nonce \(N_u\), where \(N_u\)
role session(U,V: agent,  
   Ku, Kv : public_key,  
   IDu, IDv, Vru, Vrv, Cu, Cv,  
   Sru, Srv, Tu, Tv : text,  
   H: function, F: function)
def=  
  local  SU, SV, RU, RV: channel (dy)  
  composition  
  alice(U, V, Ku, H, F, IDu, Vru, Cu, Sru, Tu, SU, RU)  
/\  bob(V, U, Kv, H, F, IDv, Vrv, Cv, Srv, Tv, SV, RV)  
end role

Figure 5.3: Role specification in HLPSL for the session.

role environment()  
def=  
  const u, v : agent,  
        ku, kv : public_key,  
        h : function,  
        f : function,  
        iu, iv, vru, vrv, cu, cv, sru, srv, tu, tv : text,  
        alice_bob_nv, bob_alice_nu, subs1, subs2 : protocol_id  
  intruder_knowledge = {u, v, h, f, ku, kv}  
  composition  
  session(u, v, ku, kv, iu, iv, vru, vrv, cu, cv, sru, srv, tu, tv, h, f)  
/\  session(v, u, ku, kv, iu, iv, vru, vrv, cu, cv, sru, srv, tu, tv, h, f)  
end role  
  goal  
  secrecy_of subs1  
  secrecy_of subs2  
  authentication_on alice_bob_nv  
  authentication_on bob_alice_nu  
end goal 
environment()  

Figure 5.4: Role specification in HLPSL for the environment and goal.

is only known to \( u \). If node \( v \) gets \( N_u \) from the message from \( u \), the node \( v \) authenticates \( u \) on \( N_u \).
5.7 Formal security verification of our scheme using AVISPA tool

- authentication_on alice.bob_nv: $v$ generates a random nonce $N_v$, where $N_v$ is only known to $v$. If node $u$ gets $N_v$ from the message from $v$, the node $u$ authenticates $v$ on $N_v$.

```
% OFMC
% Version of 2006/02/13
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
C:\progra~1\SPAN\testsuite\results\access_control.if
GOAL
as_specified
BACKEND
OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 0.01s
visitedNodes: 4 nodes
depth: 2 plies
```

Figure 5.5: The result of the analysis using OFMC of our protocol.

5.7.2 Analysis of results

We have chosen the back-end OFMC for an execution test and a bounded number of sessions model checking [14]. For the replay attack checking, the back-end checks whether the legitimate agents can execute the specified protocol by performing a search of a passive intruder. After that the back-end gives the intruder the knowledge of some normal sessions between the legitimate agents. For the Dolev-Yao model check, the back-end checks whether there is any man-in-the-middle attack possible by the intruder.

We assume that the intruder has knowledge of all public parameters including $u$, $v$, $k_u$, $k_v$ and $h(\cdot)$. We have simulated our protocol using SPAN for OFMC and CL-AtSe back-ends. The results of the analysis using OFMC and CL-AtSe are shown in Figures 5.5 and 5.6. We have obtained the test results as follows:

- OFMC reports the protocol is safe.
- CL-AtSe reports the protocol is also safe.
The formal security verification analysis of our scheme clearly shows that this scheme is secure against active attacks including replay and man-in-the-middle attacks.

![SUMMARY]
SAFE

DETAILS
BOUNDED_NUMBER_OF_SESSIONS
TYPED_MODEL

PROTOCOL
C:\progra~1\SPAN\testsuite\results\UAP_correct.if

GOAL
As Specified

BACKEND
CL-AtSe

STATISTICS
Analysed : 4 states
Reachable : 4 states
Translation: 0.01 seconds
Computation: 0.00 seconds

Figure 5.6: The result of the analysis using CL-AtSe of our protocol.

5.8 Performance comparison with related schemes

In this section, we compare the performance of our scheme with Zhou et al.’s scheme [151], Huang’s scheme [77], Kim-Lee’s scheme [85] and Huang’s scheme [78].

The description of the notations $T_{ecm}$, $T_i$, $T_h$, $T_{enc}$ and $T_{dec}$ are already provided in Section 5.6.1. We denote $T_{ecenc}$ and $T_{ecdec}$ as the time for performing public key encryption and decryption using the ECC encryption and decryption algorithms, respectively. $T_{eca}$ is the time for performing the elliptic curve point addition in $E_P(a, b)$, $T_{mul}$ is the time for executing a modular multiplication over the finite field $GF(2^{163})$ and $T_{add}$ is the time for executing a modular addition in $GF(2^{163})$.

The elliptic curve scalar multiplication and modular inverse operations are computational expensive, whereas the hashing computation is much efficient than those computations [78]. In addition, the elliptic curve encryption and decryption are also
5.8 Performance comparison with related schemes

Computationally expensive as compared to those for symmetric key encryption and decryption (for example, AES encryption and decryption). For a rough estimation of the computational complexity for different schemes, we have measured the computational cost of the schemes in terms of $T_{mul}$ as in [42], [147]. The rough estimation of different operations in terms of $T_{mul}$ are shown in Table 5.6. From this table, it is noted that the computation of an ECC point multiplication requires approximately 1200 field multiplications; an ECC point addition requires one field inversion and two field multiplications; the computation of a field inversion requires approximately three field multiplications; the computation of elliptic curve encryption and decryption require approximately 2405 and 1205 field multiplications respectively [59], [125]; and the cost of field addition is negligible. A 1024-bit modular multiplication takes 41 times longer than a field multiplication in finite field $GF(2^{163})$; AES encryption/decryption and hashing operation using SHA-1 take 0.15 and 0.36 field multiplications, respectively.

Table 5.6: Time complexity of various operations in terms of $T_{mul}$.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{ecm}$</td>
<td>$1200 T_{mul}$</td>
</tr>
<tr>
<td>$T_{eca}$</td>
<td>$5 T_{mul}$</td>
</tr>
<tr>
<td>$T_i$</td>
<td>$3 T_{mul}$</td>
</tr>
<tr>
<td>$T_{add}$</td>
<td>Negligible</td>
</tr>
<tr>
<td>$T_h$</td>
<td>$0.36 T_{mul}$</td>
</tr>
<tr>
<td>$T_{enc}$</td>
<td>$0.15 T_{mul}$</td>
</tr>
<tr>
<td>$T_{dec}$</td>
<td>$1205 T_{mul}$</td>
</tr>
<tr>
<td>$T_{ecdec}$</td>
<td>$2405 T_{mul}$</td>
</tr>
</tbody>
</table>

Table 5.7: Comparison of computational overheads among different access control schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Time Complexity</th>
<th>Rough estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhou et al. [151]</td>
<td>$3T_{ecm} + T_i + T_h + 2T_{ecdec}/T_{ecdec}$</td>
<td>$7213 T_{mul}$</td>
</tr>
<tr>
<td>Huang [77]</td>
<td>$2T_{ecm} + 4T_h$</td>
<td>$2401 T_{mul}$</td>
</tr>
<tr>
<td>Kim-Lee [85]</td>
<td>$2T_{ecm} + 9T_h$</td>
<td>$2409 T_{mul}$</td>
</tr>
<tr>
<td>Huang [78]</td>
<td>$5T_{ecm} + 9T_h$</td>
<td>$6001 T_{mul}$</td>
</tr>
<tr>
<td>Ours</td>
<td>$4T_{ecm} + T_i + 4T_h + T_{enc}/T_{dec}$</td>
<td>$4805 T_{mul}$</td>
</tr>
</tbody>
</table>

The computational cost for each sensor node to achieve authentication and establishment of a secret key with its neighbor node for our scheme and other schemes...
are given in Table 5.7. It is noted that our scheme performs better than Zhou et al.’s scheme [151] and Huang’s scheme [78], while our scheme is also comparable with that for other schemes. However, in Huang’s scheme [77] each sensor node needs to update the broadcasted hash chain after each successful authentication. Further, in Kim-Lee’s scheme [85] each node requires to update the broadcasted hash chain after each successful authentication and also to renew the hash chain.

Table 5.8: Comparison of communication costs among different access control schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$I_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhou et al. [151]</td>
<td>15232</td>
<td>21</td>
<td>20</td>
</tr>
<tr>
<td>Huang [77]</td>
<td>3904</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Kim-Lee [85]</td>
<td>4136</td>
<td>16</td>
<td>12</td>
</tr>
<tr>
<td>Huang [78]</td>
<td>3392</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Ours</td>
<td>4224</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

$I_1$: Total number of bits transmission required for messages of all phases for schemes; $I_2$: Total number of message transmissions for the scheme; $I_3$: Total number of packets transmissions during authentication and key establishment phase, and dynamic nodes addition phase for the scheme.

We have computed the total number of bits required for transmission of messages for all phases, the total number of message transmissions of all phases, and total number of packet transmissions during authentication and key establishment phase, and dynamic nodes addition phase for our scheme and other schemes. Communication costs of our scheme and other schemes are shown in Table 5.8. In wireless sensor networks, the transmission energy consumption rate is greater than the energy consumption rates for computing [154]. From the table, it is clear that Zhou et al.’s scheme [151] requires a lot of communication overhead compared with Huang’s scheme [77], Kim-Lee’s scheme [85], Huang’s scheme [78] and our scheme. Further, our scheme outperforms in term of communication overhead compared to Huang’s scheme [77], Kim-Lee’s scheme [85] and Huang’s scheme [78], because our scheme requires a few number of message transmissions only. Moreover, Huang’s scheme [77] and Kim-Lee’s scheme require the involvement of the base station during the authentication and key establishment phase, because the hash chain needs to be
broadcasted by the base station to all the existing sensor nodes in the network, whereas our scheme, Zhou et al.’s scheme [151] and Huang’s scheme [78] do not require to involve the base station during that phase.

We have then computed the number of bits required to store information prior to a node’s deployment for authentication and key establishment purpose for our scheme and other schemes. The results are given in Table 5.9. It is evident from the table that our scheme requires minimum number of bits prior to a node’s deployment in each sensor node for authentication and key establishment with its neighbor nodes as compared to that for other schemes except Huang’s scheme [77].

Table 5.9: Comparison of storage overheads among different access control schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Storage complexity required to store information prior to a node’s deployment (in bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhou et al. [151]</td>
<td>1824</td>
</tr>
<tr>
<td>Huang [77]</td>
<td>1456</td>
</tr>
<tr>
<td>Kim-Lee [85]</td>
<td>1616</td>
</tr>
<tr>
<td>Huang [78]</td>
<td>1648</td>
</tr>
<tr>
<td>Ours</td>
<td>1560</td>
</tr>
</tbody>
</table>

Table 5.10: Summary of the results of formal security verification using OFMC and AtSe model checkers for our scheme and existing schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Results using OFMC and CL-AtSe backends</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhou et al. [151]</td>
<td>Safe</td>
</tr>
<tr>
<td>Huang [77]</td>
<td>Unsafe</td>
</tr>
<tr>
<td>Kim-Lee [85]</td>
<td>Unsafe</td>
</tr>
<tr>
<td>Huang [78]</td>
<td>Unsafe</td>
</tr>
<tr>
<td>Ours</td>
<td>Safe</td>
</tr>
</tbody>
</table>

We have compared the results of the formal security verification of all schemes in Table 5.10, under AVISPA backends: OFMC and CL-AtSe. From this table, it is
clear that Zhou et al.'s scheme [151] and our scheme are secure against passive and active adversaries, while Huang's scheme [77], Kim-Lee's scheme [85] and Huang's scheme [78] are insecure against passive and active attacks including the replay and man-in-the-middle attacks.

Finally, we have considered the energy required for a sensor node during the authentication and key establishment phase in order to authenticate its neighbor node and then to establish a secret session key with that neighbor node for our scheme and other schemes. As in [57], the energy cost of a sensor node is considered due to both computational and communication costs involved during the authentication and key establishment phase. Comparison of the energy cost of a sensor node during the authentication and key establishment phase between our scheme and other schemes is shown in Table 5.11. In wireless communication, energy of sensor nodes goes mainly for transmissions and receptions of messages/packets rather than computing. Since our scheme requires less number of message or packet transmissions as compared to other schemes, the energy spent by sensor nodes is less as compared to other schemes. Due to involvement of the base station during the authentication and key establishment phase, in Huang’s scheme [77] and Kim-Lee’s scheme [85] a sensor node needs to further spend energy to receive the broadcasted information by the base station. Moreover, Huang’s scheme [77] and Kim-Lee’s scheme [85] are insecure, whereas our scheme is secure against different attacks which are shown theoretically and through simulation results using AVISPA tool. Hence, considering efficiency and security, our scheme is significantly better than the existing access control schemes.

5.9 Summary

In this chapter, we have proposed an effective novel access control scheme, which is suitable for wireless sensor networks. Using the preloaded certificate, each sensor node can authenticate and establish symmetric secret keys with its neighbor nodes. We have shown that our scheme is unconditionally secure against node capture attacks. In our scheme, the external devices are prevented from injecting false reports into the sensor network. Through the formal security analysis we have further shown that our scheme can defend different node deployment attacks by an adversary. In our scheme, we require only a few message transmissions as compared
Table 5.11: Comparison of energy cost of a sensor during authentication and key establishment phase between our scheme and other schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Sensor node’s energy cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>[151]</td>
<td>bootstrapping time validation + one hash operation for parameter generation + ECDSA signature verification to verify node’s identity, length of bootstrapping phase and node’s public key + two ECC encryption/decryption for random nonce encryption and validation + five message transmissions</td>
</tr>
<tr>
<td>[77]</td>
<td>one ECC point multiplication + one random number generation + two hash operations for parameters generation + one ECC point multiplication for computing shared session key + two hash operations for hash chain validation + four message transmissions</td>
</tr>
<tr>
<td>[85]</td>
<td>one ECC point multiplication + one random number generation + four hash operations for parameters generation + one ECC point multiplication for computing shared session key + four hash operations for hash chain validation + one hash operation for hash chain renewal + one broadcasting for hash chain renewal + four message transmissions</td>
</tr>
<tr>
<td>[78]</td>
<td>one ECC point multiplication + one random number generation + two hash operations for signature generation and verification + timestamp validation + two hash operations for parameters verification + one ECC point multiplication for computing secret shared key + three ECC point multiplication for signature verification + four message transmissions</td>
</tr>
<tr>
<td>Ours</td>
<td>two ECC point multiplication for public key computation and secret key generation + ECDSA signature verification + value of latest version verified validation, one hash operation for secret key generation, and two hash operations for puzzle message generation and validation + one symmetric-key encryption/decryption for puzzle message encryption/validation + three message transmissions</td>
</tr>
</tbody>
</table>
to that for Zhou et al.’s scheme, Huang’s scheme and Kim-Lee’s scheme, and as a result, our scheme is more appropriate for practical applications. Moreover, our scheme is efficient in computation for authentication and key establishment of symmetric secret keys with its neighbor nodes. Dynamic nodes addition in our scheme does not require any involvement of the base station during the authentication and key establishment phase as in Huang’s scheme and Kim-Lee’s scheme. Zhou et al.’s scheme requires each sensor node to sustain a tolerance time interval before it is compromised. However, our scheme, Huang’s scheme and Kim-Lee’s scheme do not have this constraint. In addition, our scheme can be easily implemented as a dynamic access control because all the old node parameters stored in each existing deployed sensor node need not be changed/updated once a new node is deployed or an old node is lost. We have also simulated our protocol for formal security analysis using AVISPA tool and it is evident from the simulation results that our protocol is secure.
Chapter 6

Certificate-Less Access Control in Distributed WSNs

In wireless sensor network, each node is preloaded with a set of keying information prior to its deployment in a target field. After deployment, using their stored information they can establish pairwise keys with each other either using the deterministic schemes or probabilistic schemes. However, most of those schemes are not scalable or they are vulnerable to a small number of captured nodes by an attacker. Further, most of these schemes can not be easily implemented as dynamic access control because all the existing old keys as well as broadcasting messages of existing nodes should be updated once new nodes are deployed in the network.

In this chapter, we propose an enhancement of Huang’s access control scheme [78] based on elliptic curve cryptosystem, which prevents malicious nodes from joining the sensor network, in the distributed wireless sensor networks. Huang’s scheme uses the expiration time of nodes so that the nodes can only authenticate with its neighbor nodes within that time. However, we show that there is a fatal weakness during the authentication procedure in Huang’s scheme, which leads to the problem that the attacker can easily intercept and distort messages such that all the legitimate nodes can be considered as the illegitimate ones by each other. In order to remedy that weakness found in Huang’s scheme, we propose a more efficient and secure access control scheme as compared with Huang’s scheme. Further, our scheme is significantly better in terms of performance and security compared with other related access control schemes. In fact, our scheme requires significantly less communication costs as compared to other related schemes. We simulate our scheme
for formal security verification using the AVISPA tool and show that our scheme is secure.

### 6.1 Notation

We use the notations given in Table 6.1 for describing an attack on Huang’s access control scheme and our proposed access control scheme. Note that a random nonce is a one-time random bit-string which is usually used to achieve freshness. The public key of base station is $Q = xG$, where $xG = G + G + \ldots + G(x$ times) = $\mathcal{O}$ is called the elliptic curve scalar multiplication in $E_q(a, b)$, $\mathcal{O}$ the point at infinity or zero point [88]. We use the secure hash function $h(\cdot)$ as SHA-1 [6].

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SN_i$</td>
<td>Identifier of sensor node $SN_i$</td>
</tr>
<tr>
<td>$E_q(a, b)$</td>
<td>An elliptic curve over finite field $GF(q)$</td>
</tr>
<tr>
<td>$G$</td>
<td>A base point on $E_p(a, b)$</td>
</tr>
<tr>
<td>$x$</td>
<td>Private key of base station</td>
</tr>
<tr>
<td>$Q$</td>
<td>$Q = xG$, public key of base station</td>
</tr>
<tr>
<td>$SK_{ij}$</td>
<td>Symmetric secret key shared between $SN_i$ and $SN_j$</td>
</tr>
<tr>
<td>$h(\cdot)$</td>
<td>Secure one way hash function</td>
</tr>
<tr>
<td>$RN_i$</td>
<td>Random nonce generated by node $SN_i$</td>
</tr>
<tr>
<td>$r_i$</td>
<td>Random number generated by base station for node $SN_i$</td>
</tr>
<tr>
<td>$DV_i$</td>
<td>Deployment version for node $SN_i$</td>
</tr>
<tr>
<td>$T_i$</td>
<td>Bootstrapping time for node $SN_i$</td>
</tr>
<tr>
<td>$A</td>
<td></td>
</tr>
<tr>
<td>$E_K(M)$</td>
<td>Symmetric key encryption of $M$ using key $K$</td>
</tr>
<tr>
<td>$D_K(M)$</td>
<td>Symmetric key decryption of $M$ using key $K$</td>
</tr>
<tr>
<td>$\mathcal{A}$</td>
<td>Adversary</td>
</tr>
</tbody>
</table>
6.2 Review of Huang’s access control scheme

In this section, we briefly review the Huang’s access control protocol [78] for wireless sensor networks based on elliptic curve cryptography.

Huang’s scheme is based on ECC and the concept of Schnorr signature [124]. This access control scheme uses the concept of time bound in which once time period has elapsed, the sensor nodes in wireless network cannot access any data for future time period. For that purpose, each node is given its own expiration time $w$. A node can achieve authentication and establishment of secret keys with other nodes in the time period $z$ if and only if $z \leq w$. Once the time period $z > w$ elapses, any node is not allowed for authentication and key establishment with other nodes. This scheme is discussed briefly in the following subsections.

6.2.1 Pre-deployment phase

In this phase, the base station first selects a secret key $x$ and computes its public key $Q = xG$ over the elliptic curve $E_q(a,b)$, where $G$ is the base point whose order is $n$ (of at least 160 bits).

It is assumed that there is a number of $v$ neighbor nodes with identities $\{SN_1, SN_2, \ldots, SN_v\}$ in a designated area. For each node $SN_i$, the base station generates a random number $r_i$, an expiration time $w_i(< t)$ and computes the public value $R_i = r_iG = (Rx_i, Ry_i)$, the value $s_i = r_i + c_i x \pmod q$, where $c_i = h(SN_i||Rx_i||Ry_i||w_i)$. The base station preloads $E_q(a,b)$, $Q$, $G$, $n$, one-way hash function $h(\cdot)$, $w_i$, and $(R_i, s_i)$ to each node $SN_i$’s memory ($i = 1, 2, \ldots, r$).

6.2.2 Authentication and key establishment phase

Let the time period be $T$. Suppose two nodes $SN_i$ and $SN_j$ want to authenticate and establish secret key between them. Node $SN_i$ first generates a random number $t_i (< n)$, computes the public value $A_i = t_iG$ and sends the message $\langle SN_i||A_i \rangle$ to node $SN_j$. Similarly, node $SN_j$ also generates a random number $t_j (< n)$, computes the public value $A_j = t_jG$ and sends the message $\langle SN_j||A_j \rangle$ to node $SN_i$.

Node $SN_i$ then computes the secret key shared with $SN_j$ as $K_{i,j} = t_iA_j = (K_{x_{ij}}, K_{y_{ij}})$ and also computes the signature $z_i = t_i + e_is_i \pmod q$, where $e_i = \ldots$. 
$h(SN_i || K_{x_{ij}} || K_{y_{ij}})$, and sends the following message to node $SN_j$:

$$SN_i \rightarrow SN_j : \langle z_i || R_i || w_i \rangle.$$ 

Similarly, node $SN_j$ also computes the same secret key shared with $SN_i$ as $K_{i,j} = t_j A_i = (K_{x_{ij}}, K_{y_{ij}})$. After that $SN_j$ computes the signature $z_j = t_j + e_j s_j \pmod{q}$, where $e_j = h(SN_j || K_{x_{ij}} || K_{y_{ij}})$, and sends the following message to node $SN_i$:

$$SN_j \rightarrow SN_i : \langle z_j || R_j || w_j \rangle.$$ 

Finally, nodes $SN_i$ and $SN_j$ verify the authenticity of each other using values of $z_i$, $z_j$, $w_i$ and $w_j$ as follows. $SN_i$ checks whether $w_j > T$ and $z_j G = A_j + e_j (R_j + c_j Q)$, where $c_j = h(SN_j || R_{x_j} || R_{y_j} || w_j)$, $e_j = h(SN_j || K_{x_{ij}} || K_{y_{ij}})$ and $R_j = (R_{x_j}, R_{y_j})$. If these hold, $SN_i$ makes sure that $SN_j$ is a legitimate. Similarly, node $SN_j$ checks whether $w_i > T$ and $z_i G = A_i + e_i (R_i + c_i Q)$, where $c_i = h(SN_i || R_{x_i} || R_{y_i} || w_i)$, $e_i = h(SN_i || K_{x_{ij}} || K_{y_{ij}})$ and $R_i = (R_{x_i}, R_{y_i})$. If these hold, $SN_j$ also makes sure that $SN_i$ is a legitimate. The exchanges of messages in this phase are summarized in Table 6.2.

Table 6.2: Authentication and key establishment phase of Huang’s scheme [78].

<table>
<thead>
<tr>
<th>Node $SN_i$</th>
<th>Node $SN_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle SN_i</td>
<td></td>
</tr>
<tr>
<td>$\langle z_i</td>
<td></td>
</tr>
</tbody>
</table>

6.2.3 Dynamic node addition phase

The same procedure is applied for a new node $N_{r+1}$ deployment. The base station needs to preload the information as done in pre-deployment phase for other nodes. After deployment, the new node performs the above authentication and key establishment phase in order to authenticate and establish pairwise secret keys with its existing neighbor nodes in the network.
6.3 An attack on Huang’s access control scheme

In this section, we show that the recently proposed Huang’s access control scheme [78] is vulnerable to a fatal weakness, that is, it is insecure against an active attack such as man-in-the-middle attack even though its authentication procedure and common key generation are simple and efficient. We show that the following man-in-the-middle attack is possible by an active attacker $A$, who has ability to eavesdrop messages, modify, delete or change the messages being transmitted over the public channel. Our attack on Huang’s scheme is summarized in Table 6.3.

The attack has the following steps:

Step 1. Node $SN_i$ first generates the private key $t_i$, computes the public key $A_i = t_iG$ over the elliptic curve $E_q(a,b)$, and sends $\langle A_i ||SN_i \rangle$ to its neighbor node $SN_j$, where $G$ is the base point of the elliptic curve $E_q(a,b)$.

Step 2. The attacker $A$ intercepts the message $\langle A_i ||SN_i \rangle$ and blocks this message. $A$ generates a private key $t_{ai}$ and computes the public key $A'_i = t_{ai}G$. $A$ then sends $\langle A'_i ||SN_i \rangle$ to node $SN_j$ by replacing $A_i$ by the modified $A'_i$ in the message.

Step 3. Similar to node $SN_i$, node $SN_j$ generates secret $t_j$, computes public $A_j = t_jG$ over the elliptic curve $E_q(a,b)$, and sends $\langle A_j ||SN_j \rangle$ to node $SN_i$.

Step 4. $A$ again intercepts the message $\langle A_j ||SN_j \rangle$ and blocks this message. $A$ generates another private $t_{aj}$, computes public $A'_j = t_{aj}G$ over the elliptic curve $E_q(a,b)$, and finally sends $\langle A'_j ||SN_j \rangle$ to node $SN_i$ by replacing $A_j$ by the modified $A'_j$ in the intercepted message.

Step 5. Node $SN_i$ generates the secret key $K_{ia} = t_iA'_j = t_it_{aj}G$ shared with node $SN_j$. However, in practice it is the secret key shared between $SN_i$ and $A$. Node $SN_j$ also generates the secret key $K_{ja} = t_jA'_i = t_jt_{ai}G$ shared with $SN_i$, whereas it is the actual key shared between $SN_j$ and $A$.

Note that $A$ has the above two keys $K_{ia} = t_{aj}A_i = t_{aj}t_iG$, which is the key shared with $SN_i$, and $K_{ja} = t_{ai}A_j = t_{ai}t_jG$, which is the key shared with $SN_j$. Thus, $A$ is able to establish two secret keys shared with $SN_i$ and $SN_j$. However, both $SN_i$ and $SN_j$ think that they are sharing only a single common secret key.
Table 6.3: An active attack on Huang’s scheme during the authentication and key establishment phase.

<table>
<thead>
<tr>
<th>Node $SN_i$</th>
<th>Adversary $A$</th>
<th>Node $SN_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate $t_i$.</td>
<td>Intercept $A_i$.</td>
<td>Generate $t_j$.</td>
</tr>
<tr>
<td>Compute $A_i = t_iG$.</td>
<td>Generate $t_{ai}$.</td>
<td>Compute $A_j = t_jG$.</td>
</tr>
<tr>
<td>$\langle A_i</td>
<td></td>
<td>SN_i\rangle$</td>
</tr>
<tr>
<td>Intercept $A_i$.</td>
<td>$\langle A'_i</td>
<td></td>
</tr>
<tr>
<td>Generate $t_{ai}$.</td>
<td>Compute $A'<em>j = t</em>{aj}G$.</td>
<td>$\langle A'_j</td>
</tr>
<tr>
<td>Compute $K_{ia} = t_iA'_j$</td>
<td>Compute $K_{ja} = t_jA'_i$</td>
<td></td>
</tr>
<tr>
<td>$= (K_{x_{ia}}, K_{x_{ia}})$</td>
<td>$= (K_{x_{ja}}, K_{x_{ja}})$</td>
<td></td>
</tr>
<tr>
<td>$z_i = t_i + e_is_i \mod q$.</td>
<td>$z_j = t_j + e_js_j \mod q$.</td>
<td></td>
</tr>
<tr>
<td>$\langle z_i</td>
<td></td>
<td>R_i</td>
</tr>
<tr>
<td>Check if $w_j &gt; T$?</td>
<td>Check if $w_i &gt; T$?</td>
<td></td>
</tr>
<tr>
<td>Verify $z_jP = A'_j$</td>
<td>Verify $z_iG = A'_i$</td>
<td></td>
</tr>
<tr>
<td>$+e_j(R_j + c_jQ)$</td>
<td>$+e_i(R_i + c_iQ)$.</td>
<td></td>
</tr>
<tr>
<td>Later verification</td>
<td>Later verification</td>
<td></td>
</tr>
<tr>
<td>will fail.</td>
<td>will fail.</td>
<td></td>
</tr>
</tbody>
</table>
6.3 An attack on Huang’s access control scheme

Step 6. $SN_i$ computes $z_i = t_i + e_is_i \mod q$, where $e_i = h(SN_i||K_{xa}||K_{ya})$, $K_{ia} = (K_{xa}, K_{ya})$, $s_i = r_i + c_ix \mod q$, $c_i = h(SN_i||R_{xi}||R_{yi}||w_i)$, $R_i = r_iG = (R_{xi}, R_{yi})$, and sends the message $(z_i||R_i||w_i)$ to node $SN_j$. Here $w_i$ is the expiration time for $SN_i$.

Step 7. $A$ does not intercept the message $(z_i||R_i||w_i)$ and it goes as it is to $SN_j$.

Step 8. $SN_j$ sends $(z_j||R_j||w_j)$ to $SN_i$, where $z_j = t_j + e_js_j \mod q$, $e_j = h(SN_j||K_{xa}||K_{ya})$, $K_{ja} = (K_{xa}, K_{ya})$, $s_j = r_j + c_jx \mod q$, $c_j = h(SN_j||R_{xj}||R_{yj}||w_j)$, $R_j = r_jG = (R_{xj}, R_{yj})$, where $w_j$ is the expiration time for $SN_j$.

Step 9. $A$ does not intercept the message $(z_j||R_j||w_j)$ and sends the same message to $SN_i$.

Step 10. In order to ensure $SN_j$ is a legitimate node and confirms their shared session key, $SN_i$ first compares $SN_j$’s expiration time $w_j$ with the broadcasted time period $T$ from the system. Since $w_j$ is not changed by the attacker $A$, the verification $w_j < T$ holds. After that $SN_i$ computes $c_j = h(SN_j||R_{xj}||R_{yj}||w_j)$ where $R_j = (R_{xj}, R_{yj})$, $e_j = h(SN_j||K_{xa}||K_{ya})$, $K_{ia} = (K_{xa}, K_{ya})$ and checks whether $z_jG = A_j' + e_j(R_j + c_jQ)$ holds or not, where $Q$ is the public key of the base station. However,

$$A_j' + e_j(R_j + c_jQ) = t_{a_j}G + e_j(r_jG + c_jxG) = (t_{a_j} + e_j(r_j + c_jx))G = (t_{a_j} + e_js_j)G \neq (t_j + e_js_j)G \neq z_jG.$$

Thus, the signature verification fails and node $SN_i$ will consider node $SN_j$ as an illegitimate node.

Step 11. Node $SN_j$ also compares $SN_i$’s expiration time $w_i$ with the broadcasted time period $T$ from the system, which holds true in this case. After that $SN_j$ computes $c_i = h(SN_i||R_{xi}||R_{yi}||w_i)$, where $R_i = (R_{xi}, R_{yi})$, $e_i = h(SN_i||K_{xj}||K_{yj})$, where $K_{ja} = (K_{xj}, K_{yj})$, and finally checks whether
\[ z_iG = A'_i + e_i(R_i + c_iQ) \] holds or not. It is again clear that

\[
A'_i + e_i(R_i + c_iQ) = t_{ia}G + e_i(r_iG + c_ixG) = (t_{ia} + e_i(r_i + c_ix))G = (t_{ia} + e_is_i)G \\
\neq (t_i + e_is_i)G \\
\neq z_iG.
\]

As a result, the signature verification also fails and node \( SN_j \) will consider node \( SN_i \) as an illegitimate node.

In our attack, the adversary (attacker) can block the correct \( A_i \) and resubmit the distorted \( A'_i \) to node \( SN_j \) after modifying the value of \( A_i \). Node \( SN_i \) will not be then able to pass the authentication in node \( SN_j \), and node \( SN_j \) considers node \( SN_i \) as an illegitimate node because the later verification equation cannot hold. On the other hand, when node \( SN_j \) communicates with node \( SN_i \), the adversary can also intercept and modify the value of \( A_j \), and resubmit the distorted value of \( A'_j \) so that the authentication of node \( SN_j \) also fails.

It is important to note that the same problem described above in the attack appears in the new node addition phase, because the authentication and key establishment procedure for any old node with a new node remains same as the authentication and key establishment procedure between old nodes. All the legitimate new nodes will also be considered as illegitimate due to the above active attack of the adversary. Then attacker can intercept and distort these messages such that all the legitimate nodes will be considered as the illegitimate ones. Hence, Huang’s scheme fails to provide strong authentication during the authentication and key establishment procedure and as a result, Huang’s scheme is insecure against man-in-the-middle attack.

### 6.4 The proposed certificate-less access control scheme

In this section, in order to remedy the weakness found in Huang’s scheme [78], we propose an enhanced access control scheme. For this purpose, we first discuss the threat model and then different phases of our scheme.
6.4 The proposed certificate-less access control scheme

6.4.1 Threat model

In applications where sensor nodes are deployed in hostile environments, sensor nodes can be physically captured by an attacker. Due to cost constraints sensor nodes are not equipped with tamper-resistant hardware and thus, once a sensor node is captured by an attacker, all the sensitive data as well as cryptographic information stored in its memory are revealed to that attacker. Using the sensitive information from captured nodes, the attacker can deploy some fake/malicious nodes in any part of the deployment field. However, we assume that in any case, the base station (BS) will not be compromised by an attacker. As in [84], we make use of the Dolev-Yao threat model [151] in which two communicating parties (nodes) communicate over an insecure channel. We adopt the similar threat model for WSNs because the channel is insecure and the end-points (sensor nodes) cannot in general be trustworthy. Finally, we assume that an attacker can eavesdrop on all traffic, inject packets, modify messages and reply old messages previously delivered.

6.4.2 Different phases

In this section, we discuss the following three phases of our scheme. The pre-deployment phase is executed by the base station in offline before deployment of sensor nodes in a particular deployment field (target field). In authentication and key establishment phase, each deployed node authenticates with its neighbors. If nodes are authenticated by each other, they establish secret keys between them. Finally, in dynamic nodes addition phase, new deployed nodes authenticate with old nodes and also establish secret keys with them.

The authentication and key establishment phase of our scheme is different from that for Huang’s scheme. In our scheme, we make use of the bootstrapping time, deployment version and latest version checked variable to distinguish old nodes from new nodes and also to detect malicious new nodes deployed after initial deployment of nodes.

Pre-deployment phase

It consists of the following steps:

Step 1: Prior to deployment of sensor nodes in the target field, the base station chooses a set of network parameters which includes (i) a finite field $GF(q)$,
where $q$ is a large odd prime of at least 160-bits; (ii) an elliptic curve $E_q(a, b)$, which consists of the set of all points of $y^2 = x^3 + ax + b \pmod{q}$ such that $a, b \in \mathbb{Z}_q = \{0, 1, 2, \ldots, q - 1\}$ are constants with $4a^3 + 27b^2 \neq 0 \pmod{q}$; (iii) a base point $G$ in $E_q(a, b)$ whose order is $n$, where $n$ is at least 160-bits such that $n > 4\sqrt{q}$ and $n$ is also prime; (iv) the base station’s private key $x \in \mathbb{Z}_{n^*}$, where $\mathbb{Z}_{n^*} = \{1, 2, \ldots, n - 1\}$; and (v) the base station’s public key $Q = xG$.

Step 2: Each deployed sensor node $SN_i$ is assigned the bootstrapping time $T_i$ and the deployment version $DV_i$. The bootstrapping time indicates when a node bootstraps itself to join the sensor network and the deployment version refers to the version of a particular deployment phase.

Step 3: For each deployed sensor node $SN_i$, the base station first generates a random number $r_i \in \mathbb{Z}_{n^*}$, and computes an elliptic curve point $R_i = r_iG = (R_{x_i}, R_{y_i})$ using ECC point multiplication. The base station then computes the signature $s_i$ based on the Schnorr signature concept for each deployed sensor node $SN_i$, where $s_i = r_i + c_ix \pmod{q}$, $c_i = h(SN_i||R_{x_i}||R_{y_i}||T_i||DV_i)$, $x$ is the secret key of the base station and $r_i$ is the secret information generated by base station for each deployed sensor node $SN_i$. It is computationally infeasible for an attacker to know the value of $x$ and $r_i$ from $Q$ and $R_i$ respectively due to difficulty of solving ECDLP problem.

Step 4: The base station preloads a set of node parameters for each sensor node $SN_i$ prior to its deployment in the target field. This set contains (i) the unique node identifier $SN_i$ of the node $SN_i$; (ii) the elliptic curve $E_q(a, b)$; (iii) the base point $G$; (iv) the base station’s public key $Q$; (v) a hash function $h(\cdot)$; (vi) $DV_i$; (vii) $T_i$; (vii) $R_i$; (viii) $s_i$; and (ix) the local variable, called the latest version checked ($lvc_i$).

The deployment version $DV_i$ of a node $SN_i$ is initialized as follows

$$DV_i = \begin{cases} 
1, & \text{if } SN_i \text{ is deployed during initial deployment phase} \\
 l, & \text{if } SN_i \text{ is deployed during } l\text{-th dynamic nodes addition phase.}
\end{cases}$$

Initially, the value of $lvc_i$ of node $SN_i$ is assigned to the value of $DV_i$. The purpose of using the deployment version is that when a new node is added into the existing network, its neighbors can verify whether the new node is legitimate or not with the help of the bootstrapping time and local variable, latest version checked.
6.4 The proposed certificate-less access control scheme

Authentication and key establishment phase

This phase is executed by each deployed sensor node in the network. In the initial deployment phase, a large number of sensor nodes are deployed in the target field. We can then deploy some new sensor nodes in the existing sensor network when some nodes exhaust their power or they are compromised by an attacker.

In this phase, after deployment each node first locates its neighbors in its communication range. After that each node will authenticate its neighbor nodes in its communication range and also establish secret pairwise symmetric keys with its neighbors after successful authentication. The authentication and key establishment procedure between two neighbor nodes $SN_i$ and $SN_j$ involves the following steps:

Step 1: $SN_i$ generates a random nonce $RN_i$ and a random secret number $t_i$ ($< n$). $t_i$ is kept secret by $SN_i$ and considered as the secret key of $SN_i$. $SN_i$ computes the public key $A_i = t_iG$ and the public value $z_i = t_i + s_i \pmod{q}$ over the elliptic curve, and sends the message $\langle SN_i||RN_i||T_i||DV_i||A_i||R_i||z_i \rangle$ to its neighbor node $SN_j$.

Step 2: After receiving the message from $SN_i$, $SN_j$ verifies the bootstrapping time $T_i$ and deployment version $DV_i$ of $SN_i$ with its own $T_j$ and $DV_j$. $SN_j$ has the following three cases to verify whether $SN_i$ is legitimate or not.

Case 1: $T_i = T_j$

If $DV_i = DV_j$, $SN_j$ ensures that node $SN_i$ is deployed during the same deployment phase. Thus, nodes $SN_i$ and $SN_j$ are both new nodes. Further, $SN_j$ verifies its own $lvc_j$ with $DV_i$ received in the message. Note that $lvc_j = 1$, when $SN_j$ is deployed during the initial deployment and $lvc_j = l$, when $SN_j$ is deployed during the $l$-th dynamic node addition deployment phase. If it holds, $SN_i$ is considered as legitimate node by $SN_j$.

Case 2: $T_j > T_i$

$SN_j$ verifies whether $DV_j > DV_i$ and $lvc_j \geq DV_i$. If both conditions are satisfied, $SN_i$ is considered as a legitimate node by $SN_j$. In such a case, $SN_i$ is considered as old node and $SN_j$ is new deployed node.

Case 3: $T_j < T_i$

In this case, $SN_j$ verifies whether $DV_j < DV_i$ and $lvc_j \leq DV_i$. If both
conditions hold, $SN_i$ is considered as a legitimate node by $SN_j$. Here $SN_j$ is considered as old node and $SN_i$ is new deployed node.

For further verification $SN_j$ computes $c_i = h(SN_i||R_{x_i}||R_{y_i}||T_i||DV_i)$. $SN_j$ then checks the equality $z_iG = A_i + (R_i + c_iQ)$, where $Q$ is the public key of the base station. Note that

$$A_i + (R_i + c_iQ) = t_iG + (r_iG + c_ixG)$$
$$= (t_i + (r_i + c_ix))G$$
$$= (t_i + s_i)G$$
$$= z_iG.$$  

If the answer is yes, then $SN_i$ is accepted as a legitimate node by the node $SN_j$. It also ensures that the value $A_i$ has not altered by any attacker. $SN_j$ generates a random secret number $t_j (< n)$, which is considered as its own secret key. $SN_j$ computes the public key $A_j = t_jG$, $K_{ij} = t_jA_i = (K_{x_{ij}}, K_{y_{ij}})$ over the elliptic curve and $z_j = t_j + e_j s_j (mod q)$, where $e_j = h(SN_j||K_{x_{ij}}||K_{y_{ij}})$. It then computes the symmetric secret key $SK_{ij}$ shared with $SN_i$ as

$$SK_{ij} = h(SN_i||SN_j||T_i||T_j||DV_i||DV_j||RN_i||RN_j$$
$$z_i||z_j||K_{x_{ij}}||K_{y_{ij}}).$$  

In order to ensure that $SN_i$ will have the same secret key, $SN_j$ makes use of the challenge-response protocol as follows. $SN_j$ creates a puzzle message, say $PM$, computes the encrypted puzzle using its computed key $SK_{ij}$ as $E_{SK_{ij}}(PM)$ and then sends the message $\langle SN_j||RN_i||RN_j ||T_j||DV_j||A_j||R_j||z_j||E_{SK_{ij}}(PM)$$||h(SK_{ij}||PM||RN_i ||RN_j||T_i ||T_j||DV_i||DV_j) \rangle$ to $SN_i$.

Step 3: After receiving the message from $SN_j$, $SN_i$ proceeds as follows. $SN_i$ first verifies the received random nonce $RN_i$ in the message with its own previously generated random nonce for authentication with node $SN_j$. If it holds, $SN_i$ executes the following three cases:

Case 1: $T_i = T_j$

In this case, the bootstrapping time $T_i$ of node $SN_i$ is equal to the bootstrapping time $T_j$ of node $SN_j$. If $DV_i = DV_j$ and $lcv_i = DV_j$, $SN_i$
6.4 The proposed certificate-less access control scheme

ensures that $SN_j$ is deployed during the same deployment phase and they are both new nodes. $SN_j$ is considered as a legitimate node by $SN_i$.

Case 2: $T_i < T_j$

$SN_i$ verifies whether $DV_i < DV_j$ and $lvc_i \leq DV_j$. If both conditions are satisfied, $SN_j$ is considered as a legitimate node by $SN_i$. In such a case, $SN_i$ is considered as old node and $SN_j$ is new deployed node.

Case 3: $T_i > T_j$

$SN_i$ verifies whether $DV_j < DV_i$ and $lvc_i \geq DV_j$. If both conditions hold, $SN_j$ is also considered as a legitimate node by $SN_i$. Here, node $SN_j$ is considered as old node and $SN_i$ is new deployed node.

$SN_i$ computes $K_{ji} = t_i A_j = (K_{x_{ji}}, K_{y_{ji}})$ and the symmetric secret key $SK_{ji}$ as

$$SK_{ji} = h(SN_i || SN_j || T_i || T_j || DV_i || DV_j || RN_i || RN_j || z_i || z_j || K_{x_{ji}} || K_{y_{ji}}).$$

Note that $K_{ji} = t_i A_j = t_i t_j G = t_j (t_i G) = t_j A_i = K_{ij}$. As a result, from Equations 6.1 and 6.2, we obtain $SK_{ji} = SK_{ij}$. For further verification, $SN_i$ computes $c_j = h(SN_j || R_{x_j} || R_{y_j} || T_j || DV_j)$ and $e_j = h(SN_j || K_{x_{ji}} || K_{y_{ji}})$. Node $SN_i$ then checks whether the equality $z_j G = A_j + e_j (R_j + c_j Q)$ holds. If it holds, $SN_j$ is accepted as a legitimate node by the node $SN_i$. It is also noted that

$$A_j + e_j (R_j + c_j Q) = t_j G + e_j (r_j P + c_j xG)$$
$$= (t_j + e_j (r_j + c_j x)) G$$
$$= (t_j + e_j s_j) G$$
$$= z_j G.$$

In order to solve the puzzle, $SN_i$ first decrypts the encrypted puzzle $E_{SK_{ij}}(PM)$ using its own computed secret key $SK_{ji}$ and then retrieves the puzzle as $PM' = D_{SK_{ji}}(E_{SK_{ij}}(PM))$. After that it computes the hash value $h(SK_{ji} || PM' || RN_i || RN_j || T_i || T_j || DV_i || DV_j)$ using the retrieved puzzle $PM'$, its own computed key $SK_{ij}$, its own previously generated random nonce $RN_i$ and its timestamp $T_i$ and deployment version $DV_i$. If this computed hash value matches
with the incoming hash value received in the message, \( SN_i \) ensures that the node \( SN_j \) shares the same secret key with it.

Finally, \( SN_i \) stores the secret key \( SK_{ij} \) in its memory for future secret communication with \( SN_j \). Similarly, \( SN_j \) also stores the same secret key \( SK_{ji} (= SK_{ij}) \) in its memory for future secret communication with \( SN_i \). It is noted that our scheme provides mutual authentication between any two neighbor nodes in the network. As soon as each node \( SN_i \) authenticates and establishes secret keys with its all neighbors nodes, it updates its local version checked variable \((lvc_i)\) as \( lvc_i = lvc_i + 1 \). The purpose of this updation is that \( SN_i \) can only authenticate and establish a secret key with a new node which is to be deployed in the next deployment phase. Since the time period for authenticating and establishing secret keys with neighbors of a node is short, it is reasonable to believe that nodes are not captured during that short time period. Thus, attacker may not be able to deploy malicious nodes with the information gathered from the captured nodes in that deployment phase due to the short time period of authentication and key establishment procedure. However, the attacker can capture nodes after this short time period of authentication and key establishment phase.

The message exchanges between two neighbor nodes \( SN_i \) and \( SN_j \) are summarized in Table 6.4.

<table>
<thead>
<tr>
<th>Node ( SN_i )</th>
<th>Node ( SN_j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \langle SN_i</td>
<td></td>
</tr>
<tr>
<td>( E_{SK_{ij}}(PM)</td>
<td></td>
</tr>
</tbody>
</table>

### Dynamic node addition phase

New node deployment in sensor networks is inevitable due to the loss of sensor nodes because after several weeks or months of operation some sensor nodes in the network may exhaust their power. Moreover, some nodes may be compromised and we need
6.5 Analysis of the proposed scheme

to replace those by new deployed nodes in the network. Assume that one or more
new nodes are to be deployed in a dynamic node addition phase.

Let a new sensor node $SN_i$ be deployed in the $l$-th dynamic node addition phase.
Prior to its deployment in the target field, using the pre-deployment phase the base
station will preload a set of node parameters. This set contains (i) the unique node
identifier $SN_i$ of the node $SN_i$; (ii) the elliptic curve $E_q(a, b)$; (iii) the base point $G$;
(iv) the base station’s public key $Q$; (v) a hash function $h(·)$; (vi) $DV_i$; (vii) $T_i$; (vii)
$R_i$; (viii) $s_i$; and (ix) the local variable, called the latest version checked $lvc_i$. Note
that the value of $lvc_i$ is initialized for this case as $lvc_i = l$. After its deployment,
$SN_i$ authenticates and establishes pairwise symmetric secret keys with its neighbor
nodes using the authentication and key establishment phase as described above.

Remark 6.1: When a node is deployed in the network, it is preloaded with the
value of $DV_i$ and $lvc_i$ (for example, $DV_i = l$ and $lvc_i = l$ if the node is deployed
during $l$-th deployment phase). It may be possible that the original deployed nodes
run out of battery and hence, we require to deploy other new nodes in the existing
network. Note that after authentication and key establishment procedure with all
neighbor nodes, each node only updates its $lvc_i$ value by $lvc_i = lvc_i + 1$ so that it
can authenticate and establish keys with the new nodes deployed in the next phase.
However, nodes never update its original $DV_i$ values. As a result, authentication
is based on checking the values of $lvc_i$ of old nodes with the values of $DV_i$ of new
nodes.

6.5 Analysis of the proposed scheme

In this section, we first analyze communication overhead, computational overhead
and storage overhead required in our scheme. We then perform security analysis of
our scheme including the formal security analysis through both mathematical anal-
ysis and simulation using AVISPA model checkers. Finally, we discuss thoroughly
preform comparison of our scheme with other existing schemes.
6.5.1 Overheads

Communication overhead

From the authentication and key establishment phase described in Section 6.4.2, it is clear that any two neighbors nodes need to exchange only two messages in order to authenticate with each other and establish a symmetric secret key between them. We show bit-wise and packet-wise communication overhead for our proposed scheme. For computing the number of packets required, we have considered CC2420 transmitter [129]. CC2420 transmitter supports a packet size of 128 bytes (1024 bits). Assume that the number of bits present in deployment version $DV_i$, bootstrapping time $T_i$, node identifier $SN_i$, random nonce $RN_i$ and hash value (using SHA-1 algorithm) are 8, 32, 16, 32 and 160, respectively. Then during the authentication and key establishment phase of our scheme, the number of bits required for first message $\langle SN_i || RN_i || T_i || DV_i || A_i || R_i || z_i \rangle$ is 888 bits, which requires only one packet. On the other hand, the second message $\langle SN_j || RN_i || RN_j || T_j || DV_j || A_j || R_j || z_j \rangle \langle E_{SK_{ij}}(PM) || h(SK_{ij}) || PM || RN_i || RN_j || T_i || T_j || DV_i || DV_j \rangle$ requires 1208 bits, which needs another two packets. We see that only three packets are required for this phase.

Computational overhead

For analyzing the computational overhead for our scheme during the authentication and key establishment phase, we have used the notations shown in Table 6.5. It is clear from our authentication and key establishment phase that in order to authenticate to each other and establish a secret key between two neighbor nodes $SN_i$ and $SN_j$, $SN_i$ requires computational cost $5T_{ecm} + 3T_h + T_{dec}$ whereas $SN_j$ requires computational cost $4T_{ecm} + 4T_h + T_{enc}$.

Storage overhead

From the pre-deployment phase discussed in Section 6.4.2, it is noted that a sensor node is preloaded with the list of node parameters prior to its deployment in the target field: the unique node identifier $SN_i$ of the node $SN_i$; the elliptic curve $E_q(a, b)$; the base point $G$; the base station’s public key $Q$; $DV_i$; $T_i$; $R_i$; $s_i$; and the latest version checked ($lvc_i$). The bit sizes of these fields are same as considered in computation for communication overhead. We have taken the number of bits
6.5 Analysis of the proposed scheme

Table 6.5: Notations used for analysis of computational overhead.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_h$</td>
<td>Time for performing a one-way hash function $h(\cdot)$ (e.g., SHA-1)</td>
</tr>
<tr>
<td>$T_{enc}$</td>
<td>Time for performing a symmetric-key encryption (AES encryption)</td>
</tr>
<tr>
<td>$T_{dec}$</td>
<td>Time for performing a symmetric-key decryption (AES decryption)</td>
</tr>
<tr>
<td>$T_{ecm}$</td>
<td>Time for performing a point-multiplication in elliptic curve $E_q(a, b)$</td>
</tr>
<tr>
<td>$T_{eca}$</td>
<td>Time for performing a point-addition in elliptic curve $E_q(a, b)$</td>
</tr>
<tr>
<td>$T_{ecenc}$</td>
<td>Time for performing an encryption using ECC</td>
</tr>
<tr>
<td>$T_{ecdec}$</td>
<td>Time for performing a decryption using ECC</td>
</tr>
<tr>
<td>$T_{mul}$</td>
<td>Time for executing a modular multiplication over $GF(2^{163})$</td>
</tr>
<tr>
<td>$T_{add}$</td>
<td>Time for executing a modular addition in $GF(2^{163})$</td>
</tr>
<tr>
<td>$T_i$</td>
<td>Time for executing a modular inversion in $GF(2^{163})$</td>
</tr>
</tbody>
</table>

required to store node identifier and $lvc_i$ as 16 bits and 8 bits, respectively. Further, we have considered $q$, $a$ and $b$ for elliptic curve $E_q(a, b)$ are of 160 bits. Using Table 6.6, the storage overhead for each node requires 1664 bits.

6.5.2 Security analysis

In this section, we show that our scheme can effectively defend the following various attacks.

False reports injection

In our scheme, once a new sensor node passes the authentication procedure successfully, it has already established shared symmetric secret keys with its neighbors. These secret keys are then used to secure communications with its neighbors. Thus, the external adversaries are prevented from injecting false reports into the sensor network.

Resilience against node capture attacks

Our scheme ensures that any two neighbor sensor nodes can establish a symmetric secret key between them for their future secure communications. Moreover, the
Table 6.6: Bit size of the parameters used for our proposed access control scheme.

<table>
<thead>
<tr>
<th>Type</th>
<th>Bit size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base point ((G)) of the elliptic curve (E_p(a,b))</td>
<td>320</td>
</tr>
<tr>
<td>Public key of the BS ((Q))</td>
<td>320</td>
</tr>
<tr>
<td>Public key of sensor node (SN_i (R_i))</td>
<td>320</td>
</tr>
<tr>
<td>Secret parameter of sensor node (SN_i (s_i))</td>
<td>160</td>
</tr>
<tr>
<td>Deployment version ((DV_i))</td>
<td>8</td>
</tr>
<tr>
<td>Node identifier ((SN_i))</td>
<td>16</td>
</tr>
<tr>
<td>Bootstrapping time ((T_i))</td>
<td>32</td>
</tr>
<tr>
<td>Hash value (using SHA-1 algorithm)</td>
<td>160</td>
</tr>
<tr>
<td>Latest version verified variable ((latest_version_verified))</td>
<td>8</td>
</tr>
</tbody>
</table>

Pairwise key between any two neighbor nodes is generated using random private keys by the nodes. Thus, the established secret keys among sensor nodes are different throughout the network. Now, if a sensor node is captured or compromised, an attacker has ability to compromise secret keys of its neighbor nodes only. As a result, the effect of a captured node does not essentially lead to compromise secure communications among non-compromised nodes in the network. In other words, even some nodes are compromised, the security of the entire sensor network is not compromised, that is, non-compromised nodes still communicate among each other with 100% security. Hence, our scheme is unconditionally secure against node capture attack.

**Man-in-the-middle attack**

Suppose an adversary intercepts the message \(⟨SN_i||RN_i||T_i||DV_i||A_i||R_i||z_i⟩\) sent by sensor node \(SN_i\) to its neighbor \(SN_j\) during the authentication and key establishment phase. Let the adversary generate a secret number \(t'_i (< n)\) and compute the public value \(A'_i = t'_iG\) over the elliptic curve. Then the adversary cannot compute \(z'_i = t'_i + s_i \mod q\) from \(z_i = t_i + s_i \mod q\), because \(t_i\) is already embedded into \(z_i\). Moreover, the adversary needs to know correctly the signature \(s_i = r_i + c_i x \mod q\). To know the value of \(s_i\), the attacker has to know the private key \(x\) of the base station and and the secret information \(r_i\) generated by base station for each de-
ployed sensor node $SN_i$. To know $x$, the adversary needs to solve $Q = xG$ knowing $G$ and $Q$, which is computationally infeasible problem due to difficulty of solving ECDLP. Also, the adversary needs to know $r_i$ from $R_i = r_iG$, which is again computationally infeasible problem due to difficulty of solving ECDLP. Thus, the adversary does not have ability to change $z_i$ and $A_i$ in the intercepted message $\langle SN_i||RN_i||DV_i||A_i||R_i||z_i \rangle$. Similarly, the adversary does not have ability to change $A_j$ and $z_j = t_j + e_j s_j \pmod{q}$ due to difficulty of solving ECDLP in the reply message $\langle SN_j||RN_i||RN_j||T_j||DV_j||A_j||R_j||z_j ||E_{SK_{ij}}(PM) ||h(SK_{ij})||PM||RN_i||RN_j||T_i||T_j||DV_i||DV_j \rangle$ sent by $SN_j$ to $SN_i$.

Moreover, the parameters $(R_i, z_i)$ can not be adjustable by any adversary, because for each deployed sensor node $SN_i$, the base station first generates a random number $r_i$, computes $R_i = r_iG = (Rx_i, Ry_i)$ and then the signature $s_i = r_i + c_i x \pmod{q}$, where $c_i = h(SN_i||Rx_i||Ry_i||T_i||DV_i)$ and $x$ is the base station’s private key. As a result, adjusting $R_i = r_iG$ is not possible by the adversary as $r_i$ is used to compute the signature $s_i$ for sensor node $SN_i$ and deriving $r_i$ from $R_i = r_iG$ is again computationally infeasible problem due to difficulty of solving ECDLP. Since the private key $x$ of the base station can not be compromised, so the modification of $s_i = r_i + c_i x \pmod{q}$ is also not possible for any adversary, and $z_i$ is computed using the value of $s_i$ and the private key $t_i$ of sensor node $SN_i$ as $z_i = t_i + s_i \pmod{q}$. Hence, modification or adjustment of $(R_i, z_i)$ by any adversary is not possible. As a result, the man-in-the-middle attack is clearly prevented in our scheme.

### Formal security analysis of the proposed scheme

In this section, we show that our enhanced access control scheme has the ability to protect different attacks. We follow the similar proof as in [20], [21], [42], [53], [112], [117], [118] for the formal security analysis of our proposed scheme. We have used the method of contradiction proof [25] for our formal security analysis. Note that one can also prove the formal security in the standard model. However, in this thesis, we have performed the formal security analysis under the generic group model of cryptography. The formal security proof for preventing different attacks is given in the following theorem.

**Theorem 6.1.** Under the elliptic curve discrete logarithm problem (ECDLP) assumption, our enhanced scheme prevents malicious node deployment attack, sybil...
attack, node replication attack and wormhole attack by an adversary.

Proof. To prove this theorem, we consider the formal definition of the elliptic curve discrete logarithm problem (ECDLP) given in Definition 5.1 (Chapter 5).

We use the method of contradiction proof [25] in proving this theorem. We assume that an adversary can determine the private key $x$ of the CA from public elliptic curve parameters and the CA’s public key $Q = xG$. The adversary then can have the ability to generate a valid $s_w = r_w + c_w x$ with proper bootstrapping time, deployment version and node identifier for a new deployed malicious node $SN_w$ in the network on behalf of the CA.

For this purpose, we now define the following two oracles for the adversary $A$:

- **RevealPrivateKey**: This unconditionally outputs the private key $k$ of the CA using the public elliptic curve parameters and the CA’s public key $Q = kG$.

- **CreateValidCertificate**: This query is allowed at any time during the attacker’s execution. This oracle computes a valid certificate of a new deployed malicious node on behalf of the CA.

The adversary $A$ can run the following experimental algorithm provided in Algorithm 5, Experiment$^{ECDLP}_{EACP,A}$, for the proposed enhanced access control protocol $EACP$. We define $Succ^{ECDLP}_{EACP,A} = 2Pr[\text{Experiment}^{ECDLP}_{EACP,A} = 1] - 1$. Then the advantage function becomes $Adv^{ECDLP}_{EACP}(t, q_R, q_C) = \max_A\{Succ^{ECDLP}_{EACP,A}\}$, where the maximum is taken over all $A$ with execution time $t$, $q_R$ is the number of queries to the RevealPrivateKey oracle and $q_C$ the number of queries to the CreateValidCertificate oracle. We say that our proposed enhanced access control scheme is secure against malicious node deployment attack, sybil attack, node replication attack and wormhole attack, if $Adv^{ECDLP}_{EACP}(t, q_R, q_C) \leq \epsilon$, for any sufficiently small $\epsilon > 0$.

According to the experiment provided in Algorithm 5, the adversary $A$ can correctly guess or compute the private key $x$ of the CA and the secret number $r_i$ from $R_i = r_iG$. Once the private key $x$ of the CA is known to the adversary $A$, he/she can easily generate a valid $s_w$ for a new malicious deployed node. Now the new deployed node can authenticate and establish successfully the secret pairwise keys with its neighbor nodes in the network easily. Thus, by including the proper bootstrapping time and version of the certificate a new malicious node can easily join to the network. However, this is a contradiction because of the ECDLP assumption.


Algorithm 5 Experiment\textsubscript{ECDLP}\textsubscript{EACP,\textsc{A}}

1: Call RevealPrivateKey oracle for getting the private key $x'$ from the public key $Q = xG$. Let $x' \leftarrow \text{RevealPrivateKey}(E_q(a,b),G,Q)$ be the output, which be the private key of the CA.

2: Call again RevealPrivateKey oracle for getting the secret value $r'_i$ from the public value $R_i = r_iG$. Let $r'_i \leftarrow \text{RevealPrivateKey}(E_q(a,b),G,R_i)$ be the output, which be the secret random number of the base station created for a node $SN_i$.

3: Call CreateValidCertificate oracle. Let $s_w \leftarrow \text{CreateValidCertificate}(x',E_q(a,b),q,r'_i) = r'_i + c_w x \mod q$ be the output, which be a certificate generated for a new malicious deployed node $w$ by the adversary $\mathcal{A}$ using the private key $x'$ and secret value $r'_i$ computed by the RevealPrivateKey oracle calls, where $c_w = h(SN_w||R_{x_i}||R_{y_i}||T_i||DV_w)$.

4: if (CreateValidCertificate oracle generates correctly a valid certificate for the node $w$) then
5: return 1 (Success)
6: else
7: return 0 (Failure)
8: end if

defined above. Computing the private key $x$ of the CA from the public elliptic curve parameters and the public key $Q = xG$ of the CA, it is computationally infeasible problem.

In a node replication attack, an attacker has ability to deploy malicious nodes which are clone of a captured node into the sensor network [120]. Suppose an attacker captures a node $SN_i$ in the network. Since $SN_i$ is not equipped with tamper-resistant hardware, the attacker can easily extract all the information stored in that sensor node from its memory. Note that the attacker knows the public base point $G$, the public key of the base station $Q = xG$, $DV_i$, $T_i$, $R_i = r_iG$, and $s_i = r_i + c_i x \mod q$, where $c_i = h(SN_i||R_{x_i}||R_{y_i}||T_i||DV_i)$, $R_i = (R_{x_i}, R_{y_i})$ and $x$ is the private key of the base station. Now, it is computationally infeasible problem to compute $r_i$ and $x$ from the public $R_i$ and $Q$ respectively due to the difficulty of ECDLP. Thus, the attacker does not have any ability to modify $SN_i$, $DV_i$, $T_i$, $R_i$ and $c_i$ to recompute $s_i$. As a result, the attacker has to use the same information for a new malicious deployed node. Let a new malicious node have been deployed in...
any part of the network. Note that other sensor nodes have already updated their latest version verified values to \( j + 1 \) during the \( j \)-th deployment phase after key establishment. When that malicious node tries to authenticate and establish secret keys with existing nodes, the existing nodes will deny because the authentication will not be successful due to verification of deployment version, bootstrapping time and latest version checked values.

In sybil attack, a malicious node claims multiple IDs (identifiers) or locations [63]. Since our scheme uses the proper bootstrapping time and deployment version, a new malicious deployed node is not allowed to join the sensor network. New malicious deployed nodes are prevented from falsifying the latest bootstrapping time and deployment version in our proposed scheme, because for computing \( z_i \) for node \( SN_i \) an attacker requires

\[
s_i = r_i + c_ix \pmod{q}, \quad c_i = h(SN_i||Rx_i||Ry_i||T_i||DV_i).
\]

Since \( x \) is the private key of the base station, it is known to the base station. Moreover, computing \( r_i \) and \( x \) from \( R_i = r_iG \) and \( Q = xG \) are computationally infeasible due to the difficulty of the elliptic curve discrete logarithm problem (ECDLP). A malicious node cannot then construct \( s_i \) and as a result, he/she cannot claim a new identity \( SN_i \) in the vicinity of node \( SN_j \).

In wormhole attack, an attacker can try to tunnel normal messages between a new node and other distant old nodes such that these nodes might be able to construct communication through handshakes. Our scheme withstand this attack due to the following reason. For each sensor node \( SN_i \) using the private key \( x \) only the base station can generate \( s_i = r_i + c_ix \pmod{q} \) and \( z_i = t_i + s_i \), where

\[
c_i = h(SN_i||Rx_i||Ry_i||T_i||DV_i) \quad \text{and} \quad R_i = (Rx_i, Ry_i)
\]

such that the signature verification \( z_iG = A_i + (R_i + c_iQ) \) by its neighbor node \( SN_j \) holds and also the signature verification \( z_jG = A_j + e_j(R_j + c_jQ) \) by \( SN_i \) holds, where

\[
s_j = r_j + c_jx \pmod{q}, \quad z_j = t_j + e_js_j \quad \text{and} \quad e_j = h(SN_j||K_{x_{ij}}||K_{y_{ij}}).
\]

However, without knowing the private key \( x \) of the base station, an attacker cannot easily forge the base station to deploy new malicious nodes in the sensor network.

As a result, due to difficulty of solving the elliptic curve discrete logarithm problem (ECDLP) we finally have \( Adv_{EACP}^{ECDLP}(t,q_R,q_C) \leq \epsilon \), for any sufficiently small \( \epsilon > 0 \), since \( Adv_{EACP}^{ECDLP}(t) \leq \epsilon \), for any sufficiently small \( \epsilon > 0 \) (see Definition 2.3, Chapter 2). Hence, our enhanced access control scheme prevents malicious node deployment attack, sybil attack, node replication attack and wormhole attack by an adversary.
6.5 Analysis of the proposed scheme

**Remark 6.2:** If a sensor node is attacked or seized by an adversary, that node cannot pass the authentication process in our scheme in order to establish secret keys with its neighbor nodes. However, it can access all the data in the memory and storage. In our scheme, the established secret keys are different throughout the network for each pair of neighbor nodes. When the other nodes start transmitting signals, the attacked node can still listen to the transmitted signals but it cannot get any sensitive information from the eavesdropped messages because those messages are already encrypted by secret keys which are not held by that adversary.

**Remark 6.3:** Our scheme offers 100% network connectivity, which means that any sensor node can establish secret keys with its all neighbors if authentication is successful. Consider a node $SN_u$ connects two old nodes $SN_i$ and $SN_j$ and the node $SN_u$ is compromised by an attacker. Since the node $SN_u$ is now attacked, nodes $SN_i$ and $SN_j$ will stop communication with each other via $SN_u$. If $SN_i$ wants to communicate securely with $SN_j$, there are two possible ways to do in our scheme. First, a new node $SN'_u$ may be deployed in place of $SN_u$ and after establishing secret keys between $SN_i$ and $SN'_u$ as well as between $SN'_u$ and $SN_j$, $SN_i$ and $SN_j$ can restart secure communication via $SN'_u$. Second, $SN_i$ and $SN_j$ can find another common neighbor $SN_v$ such that $SN_i$ and $SN_v$ share a secret key between them and $SN_v$ and $SN_j$ also share a secret key between them. In such a scenario, $SN_i$ can then easily communicate securely with $SN_j$ via the intermediate neighbor $SN_v$.

**Remark 6.4:** Let $SN_i$ and $SN_j$ be two neighbors nodes and if node $A$ can attack in the middle, i.e., eavesdrop messages, modify, delete or change the messages being transmitted over the public channel, $SN_i$ and $SN_j$ can still receive signals from each other. In our scheme, even if the node $A$ blocks the message $\langle SN_i || RN_i || T_i || DV_i || A_i || R_i || z_i \rangle$ and tries to send the modified message by the distorted values of $A_i$ and $z_i$, then $A$ needs to modify correctly the values of $A_i$ and $z_i$. However, there is no way to modify these values, because in order to modify $z_i$, the attacker needs the private key $x$ of the base station, which is computationally infeasible due to ECDLP. As a result, there is no way for $A$ to modify $A_i$. Similarly, $A$ cannot also modify $A_j$ and $z_j$ due to difficulty of solving ECDLP in the reply message $\langle SN_j || RN_j || RN_i || T_j || DV_j || A_j || R_j || z_j \| E_{SK_{ij}}(PM) \| h(SK_{ij}) || PM || RN_i || RN_j || T_i || T_j || DV_i || DV_j \rangle$ sent by $SN_j$ to $SN_i$. Hence, in such a scenario, the at-
tacked node \( A \) will not gain any advantages attacking in the middle.

6.6 Formal security verification of our scheme using AVISPA tool

In this section, we analyze the formal security of our proposed scheme using the AVISPA tool. For this purpose, we specify our scheme in HLPSL (High Level Protocols Specification Language). We then show the simulation results for the formal security analysis of our scheme.

```
role alice (SNi, SNj, BS : agent, SKij : symmetric_key, H : hash_func, TMi, Vi : text, Snid, Rcv : channel(dy)) played_by SNi

def=
  local State : nat, TMj, Vj, RNi, RNj, PM : text, Ti, Tj, Ri, Rj, Ci, Cj, X, Q, P : text
  const alice_bob_nj, bob_alice_ni, subs, sub1, sub2 : protocol_id

init State := 0

transition
1. State = 0 \& Rcv(start) =|>
   State’ := 1 \& RNi’ := new()
   \& secret([Ri, Rj, X], subs, BS)
   \& secret([Ti], sub1, SNi)
   \& secret([Tj], sub2, SNj)
   \& witness(SNi, SNj, bob_alice_ni, RNi’)

2. State = 1 \&
   State’ := 2 \& request(SNj,SNi.alice_bob_nj,RNj’)
end role
```

Figure 6.1: Role specification in HLPSL for the sensor node \( SN_i \) of our scheme.
6.6 Formal security verification of our scheme using AVISPA tool

6.6.1 Specifying our scheme

We have implemented our scheme for the authentication and key establishment phase under AVISPA model checkers. In AVISPA, we have implemented two basic roles: alice and bob, which represent the participants: the initiator node $SN_i$ and the responder, node $SN_j$, respectively. We have further specified the session and environment in our implementation. Figure 6.1 shows the specification in HLPSL language for the role of the initiator node $SN_i$. During the authentication and key establishment phase, node $SN_i$ sends the message $\langle SN_i||RN_i||T_i||DV_i||A_i||R_i||z_i \rangle$ to its neighbor node $SN_j$ with the help of the $Snd()$ operation. Here the type declaration channel ($dy$) indicates that the channel is for the Dolev-Yao threat model. The initiator node $SN_i$ receives the message $\langle SN_j ||RN_j||RN_j||T_j||DV_j||A_j||R_j||z_j \rangle \langle E_{SK_{ij}}(PM) \rangle \langle h(SK_{ij}|| PM||RN_i|| RN_j) \rangle$ from responder, node $SN_j$ with the help of the $Rcv()$ operation for mutual authentication. The intruder has the ability to intercept, analyze, and/or modify messages transmitted over

![Figure 6.2: Role specification in HLPSL for the sensor node $SN_j$ of our scheme.](image)

```plaintext
role bob (SNi,SNj,BS : agent,
    SKij : symmetric_key,
    H : hash_func,
    TMj,Vj : text,
    SND, Rcv: channel(dy))
played_by SNj

def=
    local State : nat,
    TMi, Vi, RNi, RNj, PM: text,
    Ti, Tj, Ri, Rj, Ci, Cj, X, Q, P : text
    const bob_alice_ni, alice_bob_nj,
    sub1, sub2 : protocol_id
init State := 0
transition
1. State = 0 \& Rcv(SNi. RNi'.H(Ti.H(Ri.Ci.X).Q).
    State' := 1 \& RNj' := new() \&
    PM' := new() \&
    secret({Ri, Rj, X}, subs, BS) \&
    secret({Ti}, sub1, SNi) \&
    secret({Tj}, sub2, SNj) \&
        .{PM}_SKij.H(SKij.PM.RNi'.RNj'.TMj.TMi.Vj)) \&
    witness(SNj, SNi, alice_bob_ni, RNj') \&
    request(SNi, SNj, bob_alice_ni, RNi')
end role
```
the insecure channel. After successful verification initiator node $SN_i$ accepts responder, node $SN_j$ as a legitimate node and then computes the symmetric secret key $SK_{ji}$ as $SK_{ji} = h(SN_i||SN_j||T_i||T_j||DV_i||DV_j||RN_i||RN_j||z_i||z_j||K_{x_i}||K_{y_j})$ using the symmetric secret key $K_{ji}$. Node $SN_i$ then decrypts the encrypted puzzle $E_{SK_{ij}}(PM)$ using the computed secret key $SK_{ji}$ and then retrieves the puzzle $PM$ send by responder, node $SN_j$. After that it computes the hash value $h(SK_{ji}||PM'||RN_i||RN_j||T_i||T_j||DV_i||DV_j)$ using the retrieved puzzle $PM'$, its own computed key $SK_{ji}$, its own previously generated random nonce $RN_i$ and its timestamp $T_i$ and deployment version $DV_i$. If this computed hash value matches with the incoming hash value received in the message, $SN_i$ accepts node $SN_j$ as a authenticated node and ensures that same shared secret key is established between them.

We have then implemented the role of the responder, node $SN_j$ as the role of Bob in HLPSL in Figure 6.2. After receiving the message $\langle SN_i||RN_i||T_i||DV_i||A_i||R_i||z_i \rangle$ with the help of the $Rcv()$ operation, from its neighbor node $SN_i$, node $SN_j$ verifies the bootstrapping time $T_i$ and deployment version $DV_i$ and checks the equality $z_iG = A_i + (R_i + c_iQ)$ by computing the value of $c_i$. After successful verification $SN_i$ is accepted as a legitimate node by the node $SN_j$. $SN_j$ then sends reply message $\langle SN_j \ || RN_i \ || RN_j \ || T_j \ || DV_j \ || A_j \ || R_j \ || z_j \ || E_{SK_{ij}}(PM) \ || h(SK_{ij}||PM)||RN_i||RN_j||T_i \ || T_j || DV_i || DV_j \rangle$ to $SN_i$ with the help of the $Snd()$ operation. In the roles, mutual authentication between $SN_i$ and $SN_j$ are performed.

In HLPSL specification, witness(A,B,id,E) declares for a (weak) authentication property of $A$ by $B$ on $E$, declares that agent $A$ is witness for the information $E$; this goal will be identified by the constant $id$ in the goal section. request(B,A,id,E) means for a strong authentication property of $A$ by $B$ on $E$, declares that agent $B$ requests a check of the value $E$; this goal will be identified by the constant $id$ in the goal section. The intruder is always denoted by $i$.

In Figure 6.3, we have given the specifications in HLPSL for the role of goal and environment. The role of session is given in Figure 6.4. In the session segment, all the basic roles: alice, bs and bob are instanced with concrete arguments. The top-level role (environment) contains all the global constants and a composition of one or more sessions, where the intruder may play some roles as legitimate users. The intruder also participates in the execution of protocol as a concrete session. The current version of HLPSL supports the standard authentication and secrecy goals.
In our implementation, the following five secrecy goals and two authentications are verified:

- secrecy_of_subs: It represents that $R_i$, $R_j$ and $X$ are kept secret to the base station $BS$.
- secrecy_of_sub1: It represents that $T_i$ is kept secret to the node $SN_i$.
- secrecy_of_sub2: It represents that $T_j$ is kept secret to the node $SN_j$.
- authentication_on_bob_alice_ni: $SN_i$ generates a random nonce $RN_i$, where $RN_i$ is only known to $SN_i$. If node $SN_j$ gets $RN_i$ from the message from $SN_i$, the node $SN_j$ authenticates $SN_i$ on $RN_i$.
- authentication_on_alice_bob_nj: $SN_j$ generates a random nonce $RN_j$, where $RN_j$ is only known to $SN_j$. If node $SN_i$ gets $RN_j$ from the message from $SN_j$, the node $SN_i$ authenticates $SN_j$ on $RN_j$. 
6.6.2 Analysis of results

We have chosen the back-end OFMC for an execution test and a bounded number of sessions model checking [14]. For the replay attack checking, the back-end checks whether the legitimate agents can execute the specified protocol by performing a search of a passive intruder. After that the back-end gives the intruder the knowledge of some normal sessions between the legitimate agents. For the Dolev-Yao model check, the back-end checks whether there is any man-in-the-middle attack possible by the intruder.

It is assumed that the intruder has knowledge of all public parameters. We have then simulated our scheme using AVISPA for OFMC and CL-AtSe model checkers. The results of the analysis using OFMC and CL-AtSe of our scheme are shown in Figures 6.5 and 6.6, and both simulation results ensure that our scheme is secure against active attacks including replay and man-in-the-middle attacks.

6.7 Performance comparison with related schemes

We compare the performance of our scheme with Zhou et al.’s scheme [151], Huang’s scheme [77], Kim-Lee’s scheme [85] and Huang’s scheme [78]. We have used the same notations for computational cost comparison provided in Table 6.5.

As pointed out in [78], point multiplication and modular inverse operations over an elliptic curve are computational expensive, whereas hashing computation is more
6.7 Performance comparison with related schemes

Figure 6.5: The result of the analysis using OFMC of our proposed scheme.

Figure 6.6: The result of the analysis using CL-AtSe of our proposed scheme.
efficient than those computations. Also, elliptic curve encryption and decryption are computationally expensive as compared to those for symmetric key encryption and decryptions (AES encryptions and decryptions). To have a rough estimation of the computational complexity, we measure the computational cost of different schemes in terms of $T_{\text{mul}}$ as in [147]. Rough estimation of different operations in terms of $T_{\text{mul}}$ are shown in Table 5.6 of Chapter 5.

From the analysis of [89], computing a point multiplication requires approximately 1200 field multiplications; an elliptic curve point addition requires one field inversion and two field multiplications; computing a field inversion requires approximately three field multiplications; cost of elliptic curve encryption and decryption require approximately 2405 and 1205 field multiplications respectively [59], [125]; and cost of field addition is negligible. Moreover, a 1024-bit modular multiplication takes 41 times longer than a field multiplication in finite field $GF(2^{163})$. From the results of Wong et al. [125], the speed for AES encryption and decryption, hash function using SHA-1 and 1024-bit modular multiplication show that $T_{\text{enc}} \approx 0.15T_{\text{mul}}$, $T_{\text{dec}} \approx 0.15T_{\text{mul}}$ and $T_h \approx 0.36T_{\text{mul}}$.

We have compared the computational complexity using both formulated results and rough quantitative analysis in Table 6.7. Our scheme is comparable with Zhou et al.’s scheme [151] and Huang’s scheme [78]. Though Huang’s scheme [77] and Kim-Lee’s scheme [85] computationally efficient than our scheme, they require more communication overheads shown in Table 6.7.

We have then compared the communication costs of different schemes with our scheme in Table 6.7 in terms of the total number of bits transmission required for messages of all phases for the scheme as well as the total number of packets transmissions during authentication and key establishment phase, and dynamic nodes addition phase for the scheme. In wireless sensor networks, the transmission energy consumption rate approximately over three orders of magnitude greater than the energy consumption rates for computing [154]. Zhou et al.’s scheme requires a lot of communication overhead compared with Huang’s scheme, Kim-Lee’s scheme, Huang’s new scheme and our scheme. Further, our scheme outperforms in term of communication overhead compared to both of the Huang’s scheme and Kim-Lee’s scheme, because our scheme requires a few number of packet transmissions only. Huang’s scheme [77] and Kim-Lee’s scheme require the involvement of the base station during the authentication and key establishment phase too, whereas our
Table 6.7: Performance comparison of our access control scheme with other schemes.

<table>
<thead>
<tr>
<th></th>
<th>[151]</th>
<th>[77]</th>
<th>[85]</th>
<th>[78]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>$3T_{ecm} + T_i$</td>
<td>$2T_{ecm}$</td>
<td>$2T_{ecm}$</td>
<td>$5T_{ecm}$</td>
<td>$4/5T_{ecm}$</td>
</tr>
<tr>
<td></td>
<td>$+T_h + 2T_{ccenc}$</td>
<td>$+4T_h$</td>
<td>$+9T_h$</td>
<td>$+4T_h$</td>
<td>$+3/4T_h$</td>
</tr>
<tr>
<td></td>
<td>$/ T_{ecdec}$</td>
<td></td>
<td></td>
<td></td>
<td>$T_{enc}/T_{dec}$</td>
</tr>
<tr>
<td>$I_2$</td>
<td>$7213T_{mul}$</td>
<td>$2401T_{mul}$</td>
<td>$2409T_{mul}$</td>
<td>$6001T_{mul}$</td>
<td>$5401T_{mul}$</td>
</tr>
<tr>
<td>$I_3$</td>
<td>15232</td>
<td>3904</td>
<td>4136</td>
<td>3392</td>
<td>4192</td>
</tr>
<tr>
<td>$I_4$</td>
<td>20</td>
<td>10</td>
<td>12</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>$I_5$</td>
<td>1824</td>
<td>1456</td>
<td>1616</td>
<td>1648</td>
<td>1664</td>
</tr>
</tbody>
</table>

Notes: $I_1$: Computation cost for each sensor node to achieve authentication and establish a secret common key with its neighbor node; $I_2$: Rough estimation for computation cost for each sensor node to achieve authentication and establish a secret common key with its neighbor node; $I_3$: Total number of bits transmission required for messages of all phases for the scheme; $I_4$: Total number of packets transmissions during authentication and key establishment phase, and dynamic nodes addition phase for the scheme; $I_5$: Total number of bits required for storage complexity in each sensor node due to storing information before deployment.

scheme, Huang’s scheme [78] and Zhou et al.’s scheme do not require to involve the base station during that phase.

In Table 6.7, we have computed the storage overhead in terms of the total number of bits required for storage complexity in each sensor node due to storing information before deployment in order to authenticate and establish symmetric keys with its neighbor nodes after deployment. Note that the required storage complexity of our scheme is comparable with that for Zhou et al.’s scheme, Kim-Lee’s scheme and Huang’s new scheme. Our scheme, Zhou et al.’s scheme, Kim-Lee’s scheme and Huang’s scheme [78] require little more storage complexity as compared to Huang’s scheme [77].

Finally, in Table 6.8, we have compared the results of the formal security verification of all schemes, under AVISPA model checkers. From this table, it is clear that Zhou et al.’s scheme [151] and our scheme are secure against passive and active adversaries, while Huang’s scheme [77], Kim-Lee’s scheme [85] and Huang’s scheme
are insecure against passive and active attacks including the replay and man-in-the-middle attacks. Though our scheme requires little more storage complexity as compared to Huang’s scheme [77], our scheme provides better security compared to existing schemes, because other schemes are vulnerable to different attacks. Overall, our scheme is significantly better than the existing access control schemes.

Table 6.8: Summary of the results of formal security verification using OFMC and CL-AtSe backends for our scheme and existing schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Results using OFMC and CL-AtSe backends</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhou et al. [151]</td>
<td>Safe</td>
</tr>
<tr>
<td>Huang [77]</td>
<td>Unsafe</td>
</tr>
<tr>
<td>Kim-Lee [85]</td>
<td>Unsafe</td>
</tr>
<tr>
<td>Huang [78]</td>
<td>Unsafe</td>
</tr>
<tr>
<td>Ours</td>
<td>Safe</td>
</tr>
</tbody>
</table>

6.8 Summary

In this chapter, we have proposed an enhanced access control scheme suitable for resource-constrained distributed wireless sensor networks in order to remedy security weakness found in Huang’s scheme [78]. In our scheme, the external adversaries are prevented from injecting false reports into the sensor network. Moreover, our scheme has ability to defend against node capture attack, node replication attack, sybil attack and wormhole attack. In our scheme, we require a few number of message transmissions compared with the exiting related access control schemes making our scheme is much appropriate for practical applications. Further, dynamic nodes addition in our scheme does not require any involvement of the base station during the authentication and key establishment phase as in Huang’s scheme and Kim-Lee’s scheme. Finally, our scheme can be easily implemented as a dynamic access control because all the old node parameters stored in each existing deployed sensor node need not to be changed/updated once a new node is deployed or an old node is lost.
Chapter 7

User Access Control in Hierarchical WBANs

In a wireless body area sensor network (WBAN), the low-power tiny sensor nodes are placed around a patient’s body for monitoring the patient’s body functions and neighboring environment of a patient [70], [101], [119], [155]. A typical example of WBAN is shown in Figure 1.4 (Chapter 1). With the help of WBAN, the patient’s health related information, such as temperature, respiration, heart rate, pulse oximeter, blood pressure, blood sugar, pH, etc. can be monitored remotely [11]. These health related information must be continuously processed in real time. The medical information need to be shared and accessed by various levels of users, such as healthcare staff, researchers, government agencies, and insurance companies for taking the important decisions such as clinical diagnosis and emergency medical responses about the patients [99].

The communication of health related information between sensors in a patient’s body in a WBAN and over the Internet to medical servers is strictly private and confidential [10], [90], [126], [128], [136]. Authenticated and correct medical data transmission are the essential requirements for WBAN, because false or unauthenticated medical information may lead to wrong treatments to the patients. Thus, the transmitted important information need to be collected where the information are encrypted to protect the patient’s privacy. In addition, the medical staff of a hospital who collects the data must be confident that the data is not tampered with, and indeed originates from that patient. The main challenges in WBAN are security, robustness and scalability. The size and resource constraints nature of bio-sensors
also play a crucial role in WBAN [128]. Health care staff can access data directly from the body area network of a patient after successful authentication. A survey on wireless body area network could be found in [86], [92], [119]. Scalability in terms of number of sensors and patients is an important factor in this type of network. To provide security and data privacy for WBAN, user access control is very much essential requirement.

User access control is critical to the successful operation as well as extensive adoption of wireless body area network services. The security framework for WBAN should consist of user authentication (identity verification), user authorization (access provided to user) and user accountability (monitoring activity and controlling access) in order to control user access and also to prevent different types of attacks. User access control can identify and impose different access privileges for different types of users. Generally for WBAN, different doctors, health care staffs as well as medical insurance company agents are the main users and all medical information of a particular patient are not required for all types of users. For example, only a concerned doctor can retrieve his/her patient’s data, but not the other patient’s information.

In this chapter, we consider that a WBAN, where sensor nodes should be small enough and power efficient so that their battery last for long time. Electronics of a sensor node for WBAN is designed to detect and transmit the low frequency and low amplitude physiological signals. Here the sensor node hardware requires wireless link (AM152100 IC) from AMI semiconductor used for the MICS band generation. Ameen et al. [11] pointed out the comparison between medical WBAN and general WSN, where it is clearly mentioned that both general WSN and medical WBAN have limited resources in battery, computation, and memory, whereas both follow dynamic network scale, heterogeneous devices ability and dense distribution. But for WBAN, sensors are single-function, safe, costly and quality devices and for general WSN, sensors are multi-functional, low cost, redundancy-based reliable devices. Generally, a WBAN follows small scale star network, where there is no redundancy in devices and deterministic node distribution as well as traffic are periodical and uni-directional and channel should be a specific medical channel. However, in case of a general WSN, there is a large scale hierarchical network, where redundant and random node distribution are followed. Here traffic may be uni-directional or bi-directional, and it follows generally point to point communication
7.1 Motivation

Our scheme is motivated by the following considerations. In WBAN, external parties (users) those are authorized to access data should get access as and when they demand. In order to allow authorized access of the real-time data from the sensor nodes inside WBAN to the authorized users on demand, there is a great need for user access control before allowing them to access the real-time data inside WBAN for which they are permitted. In healthcare applications, monitoring patient’s conditions by the expert doctors is very essential. Thus, real-time data sensed by the sensors in a patient’s body can be monitored directly by an authorized external user (doctor in that hospital) as and when demand is made. Based on critical and emergency situation of the patient, the doctor can take necessary action by instructing the nurses/medical staffs in the hospital for the patient. Hence, before allowing access to the sensitive and private real-time data of the patients, the external user (doctor) must be authenticated for a particular access privilege by the base station (medical server) as well as sensor node in the network. Considering these points, the user access control in WBAN for healthcare applications becomes a prominent research field.

7.2 Threat model

As in [58], we apply the Dolev-Yao threat model [61], in which two communicating parties (nodes) communicate over an insecure channel, for our scheme. We adopt the similar threat model for our scheme, where the channel is insecure and the endpoints (sensor nodes) cannot in general be trustworthy. Finally, we assume that an attacker can eavesdrop on all traffic, inject packets and reply old messages previously delivered. In our scheme, the base station (medical server) is assumed to be trusted and it will never be compromised by an attacker. Due to cost constraints, sensors are not equipped with tamper resistant hardware. Thus, if an attacker compromises any sensor from a patient’s body, he/she can exact all cryptographic information including the key materials, data and code stored on that node. As in [57], we assume that the compromised (captured) nodes can be detected and as a result, the base
station (medical server), users and sensor nodes know the ids of the compromised nodes. Consequently, the base station (medical server) alerts the users with the compromised sensor nodes in the network.

### 7.3 Our contributions

In this chapter, we propose a new password and group-based user access control scheme in wireless body area networks for health care applications. Our scheme has the following important properties:

- It provides password and group-based user authentication depending on the access rights provided for the genuine users in WBANs.

- It provides better security as compared with the other related user access control schemes, since it supports mutual authentication between the user and the base station and sensor node, resists denial-of-service attack, privileged-insider attack, smart card breach attack and man-in-the-middle attack.

- It supports dynamic node addition after initial deployment of nodes in the network. It will also support new node deployment for new patients and for that our proposed scheme does not require to update information in the user’s smart card.

- It supports changing the user’s password locally without the help of the base station (medical server).

- It establishes a secret session key between the user and a sensor node so that that key can be used for future secret communication of the real-time data between them inside WBAN.

- Finally, through the formal security verification using the AVISPA tool we show that our scheme is also secure against passive and active attacks, such as replay and man-in-the-middle attacks.
7.4 The proposed user access control scheme

In WSNs, specially in case of WBAN, for accessing of medical data, there exist different groups of users. In medical applications, different types of information belonging to various security levels can be generated by all kinds of sensors. With the proper access privilege, selected types of the authorized users should access proper data. This means that accessibility of a particular type of data to users is based solely on necessity.

In WSNs, a user access control provides a genuine user to get the permission for accessing the network. However, in real life scenario for WBAN, all users should not be given the same privilege for accessing the network. A particular user can access only those information which are required to him/her. To provide the user access control for WBANs, we propose a new access control scheme, where we use an access list for the users, which is composed of user identity, user access privilege mask and access group id $G_{id}$ for each user.

An access group is a collection of users with a given set of permissions assigned to the group. We have divided all the authorized users into different access groups, where each access group has right to access the data according to the privileges given to that particular group so that the multiple users having similar access permissions can be organized into the same group. The user group id, $G_{id}$ is a unique number used to identify a particular access group. The information stored in the sensor nodes are divided into multiple group access privilege levels. The user access privilege mask, $APM$ is responsible to define the user’s access privilege to the system information. $APM$ is a binary number in which each bit represents a specific information or service that can be accessed by the authenticated user under each access group. $APM$ is thus a bitmap, for example, if the first bit of access privilege mask represents the ‘heart rate’ parameter, an ‘1’ in this bit indicates that the ‘heart rate’ parameter is available for all members of this group. A user access list is defined for all the authorized users, which is composed of the user identity $U_j$, user access privilege mask $APM$ and access group id $G_{id}$ for each user $U_j$. The user identity $U_j$ is a unique number to identify a particular user. The base station is responsible to assign different user access groups based on the user tasks and access privileges.

In our scheme, a sensor node stores and processes information, and then sends those partially processed information to the next level. An authenticated user with
a lower privilege is not allowed to access the higher privilege information [139]. Depending on the probable user query, the base station prepares the group-based user access privilege mask, \( APM \) and also prepares an access list consists of the access privilege mask and respective access group identity \( G_{id} \). For accessing WBAN, the users first apply to the base station for the access permission. The base station keeps a user access list pool and associated user identifications. The access list defines the respective user’s access permission so that only the authorized users belonging to different access groups (for example, doctors, nurse, medical insurance team, patient party, etc.) can access the real-time data for monitoring a patient’s condition from the sensors inside WBAN. An example of a user access list is shown Figure 7.1.

![User Access Privilege mask](image)

**Figure 7.1:** An example of user access list (Source: [139]).

### 7.4.1 Notations

We use the notations in this chapter to describe our proposed scheme given in Table 7.1. The public key of the base station is \( K_{BS} = xG \) and \( x \) is the private key of the base station. \( G_{id,j} \) is the group id of a user \( U_j \), \( APM_j \) the access privilege mask of user \( U_j \), \( K_{BS} \) the public key of base station, \( MK_{Sn_i} \) the master key of sensor node \( SN_i \), \( RM_{uj} \) the random number for user \( U_j \) and \( K_i \) the secret key of node \( SN_i \) shared with BS.

### 7.4.2 Different phases

In this section, we discuss our proposed user access control scheme. Our scheme consists of the phases: pre-deployment phase, post-deployment phase, registration phase, login phase, authentication phase, password change phase and dynamic node addition phase, which are described in the following subsections.
7.4 The proposed user access control scheme

Table 7.1: Notations used in the proposed scheme.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SN_i$</td>
<td>Identifier of sensor node $i$</td>
</tr>
<tr>
<td>$U_j$</td>
<td>$j$-th user</td>
</tr>
<tr>
<td>$BS$</td>
<td>Base station</td>
</tr>
<tr>
<td>$PW_j$</td>
<td>Password of user $U_j$</td>
</tr>
<tr>
<td>$G_{id_j}$</td>
<td>Group id of user $U_j$</td>
</tr>
<tr>
<td>$APM_j$</td>
<td>Access privilege mask of user $U_j$</td>
</tr>
<tr>
<td>$x$</td>
<td>Private key of base station</td>
</tr>
<tr>
<td>$K_{BS}$</td>
<td>Public key of base station</td>
</tr>
<tr>
<td>$MK_{S_i}$</td>
<td>Master key of sensor node $SN_i$</td>
</tr>
<tr>
<td>$RM_{u_j}$</td>
<td>Random number for user $U_j$</td>
</tr>
<tr>
<td>$K_i$</td>
<td>Secret key of node $SN_i$ shared with BS</td>
</tr>
<tr>
<td>$h(\cdot)$</td>
<td>Secure one-way collision-resistant hash function</td>
</tr>
<tr>
<td>$T_i$</td>
<td>Bootstrapping time for node $SN_i$</td>
</tr>
<tr>
<td>$\Delta T$</td>
<td>Maximum transmission delay or preset acceptable delay threshold or expected time interval for the transmission delay</td>
</tr>
<tr>
<td>$A</td>
<td></td>
</tr>
<tr>
<td>$E_K(M)$</td>
<td>Symmetric encryption using the key $K$</td>
</tr>
<tr>
<td>$D_K(M)$</td>
<td>Symmetric decryption using the key $K$</td>
</tr>
<tr>
<td>$X \rightarrow Y : M$</td>
<td>Entity $X$ sends message $M$ to entity $Y$</td>
</tr>
</tbody>
</table>

Pre-deployment phase

This phase is used to preload the keying materials to all sensor nodes prior to their deployment. It is performed by the (key) setup server in offline. The setup server in our scheme is the base station (the medical server). This phase is performed by the base station in offline prior to deployment of sensor nodes in a patient’s body (target field). The pre-deployment phase consists of the following steps:

Step 1: The base station chooses a set of the following network parameters which includes a finite field $GF(p)$, where $p$ is a large odd prime of at least 160-bits; an elliptic curve $E_p(a, b)$, which is the set of all points of $y^2 = x^3 +$
\[ ax + b \pmod{p} \] such that \( a, b \in \mathbb{Z}_p = \{0, 1, 2, \ldots, p - 1\} \) are constants with \( 4a^3 + 27b^2 \neq 0 \pmod{p} \); a base point \( G \) in \( E_p(a, b) \) whose order is \( n \), where \( n \) is a prime and at least 160-bits such that \( n > 4\sqrt{p} \). The base station first chooses a random number as its own private key \( x \in \mathbb{Z}_n^* \), where \( \mathbb{Z}_n^* = \{1, 2, \ldots, n-1\} \). Then the base station computes its public key \( K_{BS} = xG \). Depending on the probable user query, the base station prepares the group-based user access privilege mask (APM) and prepares an access list consists of the access privilege mask and respective access group identity \( G_{id} \).

For each deployed sensor node \( SN_i \), the base station assigns a unique identifier, say \( SN_i \). The base station also assigns a unique randomly generated master key, say \( MK_{Si} \) for each deployed sensor node \( SN_i \), which is shared with the base station only. The base station computes \( x_iG = (x_{1i}, y_{1i}) \) for each sensor node \( SN_i \), where \( x_i \) is the private key for sensor node \( SN_i \) which is known to the BS. The base station then computes the secret key \( K_i = x_{1i} \pmod{p} \) for each sensor node \( SN_i \), where \( x_{1i} \) represents the \( x \)-coordinate of the ECC point \( x_iG \). For the security reasons, \( p \) is considered as a 160-bit prime number for ECC. Note that \( K_i \) is also a 160-bit number. However, to use \( K_i \) as the secret key for symmetric key encryption (for example, Advanced Encryption Standard (AES) [1]) we can use only 128 bits from 160-bits of \( K_i \).

Step 2: Once the set of network parameters are selected, the base station (BS) loads the following information into the memory of each sensor node \( SN_i \) prior to its deployment in offline: (i) a unique node identifier \( SN_i \); (ii) the elliptic curve \( E_p(a, b) \); (iii) the base point \( G \); (iv) the secret key \( K_i \) with \( x_i \); (v) the base station’s public key \( K_{BS} \); (vi) a secure one-way hash function \( h(\cdot) \); and (vii) its own master key \( MK_{Si} \).

**Post-deployment phase**

This phase helps the sensor nodes and the base station to establish secure connection between them. As soon as sensor nodes are deployed, their first task is to locate the physical neighbors within their communication ranges. For secure communication between sensor nodes, nodes require to establish pair wise secret keys between them. Since our main focus in this chapter is to address the user access control problem, therefore we assume that nodes in a WBAN can establish secret keys using some
7.4 The proposed user access control scheme

existing key establishment schemes. For example, we can use the unconditionally secure key establishment scheme [36] for pair wise key establishment between nodes in each cluster. As our main focus is on how authorized user belongs to different groups (doctors, nurse, medical insurance team, patient party, etc.) can access the real-time data for monitoring a patient’s condition from the sensors inside WBAN, so we require the secure communication between sensor nodes, and authorized users.

After deployment, each sensor node sends a message with its node identity \( SN_i \), bootstrapping time \( T_i \), and encrypted information containing \( K_i, SN_i \) and \( T_i \) to the base station:

\[
SN_i \rightarrow BS : \langle SN_i||T_i||E_{MK_{Si}}(K_i||SN_i||T_i) \rangle.
\]

After receiving the message from the sensor node \( SN_i \), the BS decrypts \( E_{MK_{Si}}(K_i||SN_i||T_i) \) with the master key \( MK_{Si} \) of \( SN_i \), and then checks the validity of received information \( K_i, SN_i, \) and \( T_i \). Note that \( T_i \) is the bootstrapping time of the sensor node \( SN_i \). The BS further checks if \(|T_i - T_{i}^*| \leq \Delta T_i\), where \( T_{i}^* \) is the current system timestamp of the BS and \( \Delta T_i \) is the maximum transmission delay or preset acceptable delay threshold or expected time interval for the transmission delay. Now, if it holds, the BS then stores \( K_i \) and \( T_i \) for the sensor node \( SN_i \).

Registration phase

In the registration phase, a user \( U_j \) needs to register with the base station for accessing the real-time data from a specific sensor node in WBAN. This phase consists of the following steps:

Step 1: User selects his/her identity \( U_j \), password \( PW_j \), his/her access group id \( G_{id_j} \) depending on his/her access privilege, and a random number \( RM_{u_j} \). \( U_j \) generates another secret random value \( N_j \), which is kept secret to \( U_j \) only.

After that \( U_j \) computes the masked password \( RPW_j = h(N_j||PW_j) \) and sends the message \( \langle U_j||RPW_j||G_{id_j}||RM_{u_j} \rangle \) to the BS via a secure channel.

Step 2: After receiving the information, the BS then calculates the secret shared hash value \( R_{U_j} = h(RPW_j||G_{id_j}||RM_{u_j}) \) for the user \( U_j \).

Step 3: Finally, the BS generates a tamper-proof smart card for the user \( U_j \) with the following parameters, and sends the smart card to \( U_j \) via a secure channel:

\[
BS \rightarrow U_j : \langle \text{SmartCard}(U_j||RM_{u_j}||h(\cdot)||RPW_j||G_{id_j}||R_{U_j}) \rangle.
\]
At the end, the BS stores $R_{U_j}$, $G_{id_j}$ and $APM_j$ for the user $U_j$. This registration phase is summarized in Table 7.2.

Table 7.2: The registration phase of our proposed scheme.

<table>
<thead>
<tr>
<th>User ($U_j$)</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selects $U_j$ and $PW_j$, $N_j$, $G_{id_j}$ and $RM_{u_j}$. Computes $RPW_j = h(N_j</td>
<td></td>
</tr>
</tbody>
</table>

Login phase

The purpose of this phase is to login to the system by a user who wants to access the real-time data from a specified sensor node in WBAN. The user $U_j$ needs to perform the following steps:

Step 1: At login, the user $U_j$ inserts his/her smart card into the card reader of a specific terminal, and then inputs his/her user id $U_j$, secret value $N_j$ and password $PW_j$ as well as his/her access group id $G_{id_j}$. The smart card then computes the masked password $RPW'_j = h(N_j||PW_j)$ and the hash value $R'_{U_j} = h(RPW'_j||G_{id_j}||RM_{u_j})$ for the user $U_j$ using the stored values of $G_{id_j}$, $RM_{u_j}$ in the smart card. The smart card checks whether $R'_{U_j} = R_{U_j}$. If this verification does not hold, $U_j$ has entered his/her password incorrectly and the phase terminates immediately. Otherwise, the smart card computes the hash value $h(R_{U_j}||T_1)$ using the timestamp $T_1$ of the system and then sends the following message to the BS:

$U_j \rightarrow BS : \langle U_j||h(R_{U_j}||T_1)||T_1 \rangle$.  

Step 2: After receiving the message in Step 1, the BS checks whether the condition $|T_1 - T_1^*| \leq \Delta T_1$ is valid or not, where $T_1$ is the timestamp of the user's
7.4 The proposed user access control scheme

system, and \( T_1^* \) is the current timestamp of the BS and \( \Delta T_1 \) is the maximum transmission delay or preset acceptable delay threshold or expected time interval for the transmission delay. If it is valid, the BS computes the hash value \( h(R_{U_j}||T_1) \) using the received timestamp \( T_1 \) and the previously computed value of \( R_{U_j} \) by the BS. After that the BS compares this computed hash value with the received hash value \( h(R_{U_j}||T_1) \) in the message. After successful matching BS computes the secret parameter \( S_j \).

The BS stores \( G_{id_j} \) and \( APM_j \) for a user \( U_j \), and each bit of the user access privilege mask \( APM_j \) represents a specific information or service, which can be accessed by the user \( U_j \). Each sensor node stores and processes the specific information and then sends those partially processed information to the next level. So, based on the user access permission the BS computes the secret parameter \( S_j = x + x_iR_{U_j} \pmod{p} \) and the hash value \( K_{U_j} = h(SN_i||U_j||K_{BS}||K_i) \) for an accessed sensor node \( SN_i \), and the user \( U_j \). Then the BS further computes the shared secret symmetric key \( UK_j = h(R_{U_j}||U_j||T_1||T_2) \) with the user \( U_j \) and sends the message \( \langle E_{UK}(SN_i||S_j||Z_j||K_{U_j})||T_2||T_1 \rangle \) to the user \( U_j \):

\[
BS \rightarrow U_j : \langle E_{UK_i}(SN_i||S_j||Z_j||K_{U_j})||T_2||T_1 \rangle.
\]

Step 3: After receiving the message in Step 2 from the BS, the user \( U_j \) verifies whether \( |T_2 - T_2^*| \leq \Delta T_2 \) is valid or not, where \( T_2 \) is the timestamp of the BS, \( T_2^* \) the timestamp of the user’s system and \( \Delta T_2 \) the maximum transmission delay or preset acceptable delay threshold or expected time interval for the transmission delay. \( U_j \) also checks the received value of \( T_1 \) with its previous \( T_1 \). If they match, it computes the same symmetric key \( UK_j \) shared with the BS with the received value of \( T_1 \), \( T_2 \) as \( UK_j = h(R_{U_j}||U_j||T_1||T_2) \), and decrypts \( E_{UK_i}(SN_i||S_j||Z_j||K_{U_j}) \) to retrieve \( S_j \), \( Z_j \) and \( K_{U_j} \). \( U_j \) then stores these retrieved values \( S_j \), \( Z_j \) and \( K_{U_j} \) for authorization purpose with the sensor node \( SN_j \).

Step 4: The BS computes two encrypted messages \( E_{MK_{S_i}}(SN_i||U_j||(APM_j \oplus G_{id_j})||R_{U_j}||T_1||T_2) \) using the master key \( MK_{S_i} \) of the sensor node \( SN_i \) and \( E_{K_i}(SN_i||U_j||G_{id_j}||T_1) \) using the key \( K_i \). The BS sends the following message.
to the sensor node $SN_i$:

$$BS \rightarrow SN_i : \langle SN_i||U_j||E_{MK_{Si}}(SN_i||U_j||(APM_j \oplus G_{id_j})||R_{U_j}||T_1||T_2)||E_{K_i}(SN_i||U_j||G_{id_j}||T_1)\rangle.$$  

Here $APM_j$ is the access privilege mask for the access group id $G_{id_j}$ for the user $U_j$.

Step 5: When the sensor node $SN_i$ receives the message in Step 4, it decrypts $E_{MK_{Si}}(SN_i||U_j||(APM_j \oplus G_{id_j})||R_{U_j}||T_1||T_2)$ using its own master key $MK_{Si}$ to retrieve the information $SN_i, U_j, (APM_j \oplus G_{id_j}), R_{U_j}, T_1,$ and $T_2$. $SN_i$ then checks received $SN_i, U_j$ and $T_2$ by checking the condition $|T_2 - T_2^*| \leq \Delta T_2$, where $T_2$ is the timestamp of the base station, $T_2^*$ the timestamp of the sensor node $SN_i$ and $\Delta T_2$ the maximum transmission delay or preset acceptable delay threshold or expected time interval for the transmission delay. If all these conditions are satisfied, $SN_i$ further decrypts $E_{K_i}(SN_i||U_j||G_{id_j}||T_1)$ using the stored key $K_i$ to retrieve the information $SN_i, U_j, G_{id_j}, T_1$. Using $G_{id_j}$, $SN_i$ computes $APM_j = (APM_j \oplus G_{id_j}) \oplus G_{id_j}$. $SN_i$ finally saves $R_{U_j}, T_1, T_2, G_{id_j},$ and $APM_j$ for authentication purpose.

Note that if we have two numbers of different lengths, we adopt the following strategy [110]. We first pad the smaller of the two numbers with leading zeros until it has the same length as the larger number. Then the two numbers are bitwise XORed using the the normal bitwise XOR procedure. Thus, if $APM$ and $G_{id}$ are of 64-bit and 8-bit, respectively, then we first pad $G_{id}$ with leading 56 zeros to make it of length 64 bits. After that $APM$ and the padded $G_{id}$ are bitwise XORed to obtain the 64-bit result.

The login phase of our scheme is summarized in Table 7.3.

**Authentication phase**

The authentication phase is required in order to authenticate the user when he/she wants to access real-time data inside WBAN.

During the login phase, when the user $U_j$ receives the message in Step 2, $U_j$ saves the values $S_j, Z_j$ and $K_{U_j}$ for authorization purpose with the sensor node $SN_i$ in Step 3.
Table 7.3: The login phase of our proposed scheme.

<table>
<thead>
<tr>
<th>User ($U_j$)</th>
<th>BS</th>
<th>Sensor node ($SN_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inserts smart card.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enters password $PW_j$.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>access group id $G_{id_j}$.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>and random number $RM_{uj}$.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computes $RPW_j' = h(N_j</td>
<td></td>
<td>PW_j)$,</td>
</tr>
<tr>
<td>$R_{Uj}' = h(RPW_j'</td>
<td></td>
<td>G_{id_j}</td>
</tr>
<tr>
<td>Checks $R_{Uj}' = R_{Uj}$.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>If it is correct, then sends</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\langle U_j</td>
<td></td>
<td>h(R_{Uj}'</td>
</tr>
<tr>
<td>Checks $</td>
<td>T_1 - T_1^*</td>
<td>\leq \Delta T_1$.</td>
</tr>
<tr>
<td>Checks $h(R_{Uj}'</td>
<td></td>
<td>T_1)$.</td>
</tr>
<tr>
<td>Computes $UK_j, S_j, Z_j, K_{Uj}$.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\langle E_{UK_j}(SN_i</td>
<td></td>
<td>S_j</td>
</tr>
<tr>
<td>Checks $</td>
<td>T_2 - T_2^*</td>
<td>\leq \Delta T_2$.</td>
</tr>
<tr>
<td>Decrypts the encrypted part.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saves $S_j, Z_j, K_{Uj}$.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>for authentication purpose.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_{MK_{S_j}}(SN_i</td>
<td></td>
<td>U_j</td>
</tr>
<tr>
<td>$E_{K_i}(SN_i</td>
<td></td>
<td>U_j</td>
</tr>
<tr>
<td>$\langle SN_i</td>
<td></td>
<td>U_j</td>
</tr>
<tr>
<td>$E_{K_i}(SN_i</td>
<td></td>
<td>U_j</td>
</tr>
<tr>
<td>Decrypts the encrypted parts.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checks $</td>
<td>T_2 - T_2^*</td>
<td>\leq \Delta T_2$.</td>
</tr>
<tr>
<td>Saves $R_{Uj}, T_1, T_2, G_{id_j}$ and $APM_j$ for authentication purpose.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Step 1: For authentication, the user $U_j$ computes the encrypted value $E_{K_{Uj}}(SN_i || U_j \ || R_{Uj} \ || G_{id_j} \ || T_1 \ || S_j \ || Z_j)$ and the hash value $h(T_1 \ || S_j \ || Z_j)$, and then sends the following authentication request message to the sensor node $SN_i$:

$$U_j \rightarrow SN_i : \langle SN_i || U_j \ || E_{K_{Uj}}(SN_i || U_j \ || R_{Uj} \ || G_{id_j} \ || T_1 \ || S_j \ || Z_j) \ || h(T_1 \ || S_j \ || Z_j) \rangle.$$  

Step 2: After receiving the authentication request message from the user $U_j$ in Step 1, the sensor node $SN_i$ performs the following in order to verify whether the user $U_j$ is legitimate or not. $SN_i$ first computes the key $K_{Uj}' = h(SN_i || U_j || K_{BS} || K_i)$ using the stored parameters and the received user id $U_j$. Then using the computed key $K_{Uj}'$, $SN_i$ decrypts $E_{K_{Uj}}(SN_i || U_j \ || R_{Uj} \ || G_{id_j} \ ||$
172 User Access Control in Hierarchical WBANs

Step 3: \( SN_i \) checks the following signature verification equation \( Z_j = S_j G \pmod{p} \). Note that

\[
Z_j = (K_{BS} + x_i GR_{U_j}) \\
= (xG + x_i GR_{U_j}) \\
= (x + x_i R_{U_j})G \\
= S_j G.
\]

Thus, if the signature verification fails, \( SN_i \) considers the user \( U_j \) as an illegal user and the phase terminates immediately. Otherwise, the sensor node \( SN_i \) checks the received \( G_{id_j} \) with the value received from the BS during the login phase. If it is satisfied, \( SN_i \) computes a secret session key \( SK_{ij} \) shared with the user \( U_j \) as \( SK_{ij} = h(SN_i||U_j||APM_j||G_{id_j}||S_j||R_{U_j}||T_1||T_2) \). Finally, \( SN_i \) sends an acknowledgment to the user \( U_j \) and the BS, and responds to the query of the user \( U_j \) depending upon the access privilege mask \( APM_j \) stored for user \( U_j \) using the secret session key \( SK_{ij} \).

Step 4: After receiving the acknowledgment from \( SN_i \), the user \( U_j \) computes the same secret session key \( SK_{ij} \) shared with the sensor node \( SN_i \) using its previous system timestamp \( T_1 \), stored \( T_2, S_j, R_{U_j} \) as \( SK_{ij} = h(SN_i ||U_j||APM_j||G_{id_j}||S_j||R_{U_j}||T_1||T_2) \). Thus, both the user \( U_j \) and the sensor node \( SN_i \) will communicate securely in future using the derived secret session key \( SK_{ij} \).

At the end of this phase, \( SN_i \) deletes \( R_{U_j}, T_1, T_2, G_{id_j}, \) and \( APM_j \) from its memory for security reasons. The user \( U_j \) also deletes \( S_j \) and \( Z_j \). This authentication phase of our scheme is summarized in Table 7.4.
Table 7.4: Authentication phase of our proposed scheme.

<table>
<thead>
<tr>
<th>User ($U_j$)</th>
<th>Sensor node ($SN_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(SN_i</td>
<td></td>
</tr>
</tbody>
</table>

Password change phase

In this phase, for security reasons a user $U_j$ may change his/her password freely and completely locally without contacting the BS. This phase consists of the following steps:

Step 1: $U_j$ inputs his/her smart card into the card reader of a specific terminal and provides his/her old password $PW_{old}^j$ and secret number $N_{old}^j$ as well as new changed password $PW_{new}^j$ and new secret number $N_{new}^j$.

Step 2: After that the smart card computes the masked old password of the user $U_j$ as $RPW_{old}^j = h(N_{old}^j || PW_{old}^j)$ and compares this value with the stored value of $RPW_j$ in the smart card. If they do not match, this means that the user $U_j$ has entered his/her old password incorrectly and hence, the password change phase terminates immediately. Otherwise, the smart card computes the hash value $R_{U_j}^{old} = h(RPW_{old}^j || G_id_j || RM_{u_j})$ with the old masked password $RPW_{old}^j$, group identity $G_id_j$ and random number $RM_{u_j}$. The smart card then further compares this computed hash value $R_{U_j}^{old}$ with the stored value of $R_{U_j}$. If they match, the smart card executes Step 3.

Step 3: The smart card then computes the new masked password $RPW_{new}^j = h(N_{new}^j || PW_{new}^j)$ and $R_{U_j}^{new} = h(RPW_{new}^j || G_id_j || RM_{u_j})$.

Step 4: Finally, the smart card replaces the old masked password $RPW_j$ with the
new masked password $RPW_j^{new}$ and the old hash value $R_{U_j}$ with the new hash value $R_{U_j}^{new}$ into the memory of the smart card.

### Dynamic nodes addition phase

New node deployment in sensor networks is inevitable due to the loss of sensor nodes because after several weeks or months of operation some sensor nodes in the network may exhaust their power. Even some nodes may be compromised and so we need to deploy new nodes in the network. Assume that one or more nodes to be deployed in a dynamic node addition phase.

Let a new sensor node $u$ be deployed during the dynamic node addition phase. Prior to its deployment, during the pre-deployment phase, the BS will preload a set of node parameters in offline. This set contains (i) a unique node identifier $SN_u$ of the node $u$; (ii) the elliptic curve $E_p(a,b)$; (iii) the base point $G$ in $E_p(a,b)$; (iv) the secret key $K_u$ with $x_u$ for node $SN_u$, where $x_u$ is the private-key of $SN_u$ and $x_uG = (x_{1u}, y_{1u})$ with $K_u = x_{1u} \pmod{p}$; (v) the base station’s public key $K_{BS}$; (vi) a hash function $h(\cdot)$; (vii) its own master key $MK_{Su}$.

After deployment, $SN_u$ sends a message containing its own identity $SN_u$, the bootstrapping time $T_u$, and the encrypted information $E_{MK_{Su}}(K_u||SN_u||T_u)$ using the master key $MK_{Su}$ to the BS:

$$SN_u \rightarrow BS : \langle SN_u || T_u || E_{MK_{Su}}(K_u||SN_u||T_u) \rangle.$$  

After deployment, $SN_u$ establishes pairwise keys between them in WBAN using [36]. Then $SN_u$ authenticates and establishes pairwise symmetric secret keys with the user $U_j$ as described in the login and authentication phases. Thus, in our scheme, dynamic nodes addition phase is simple and efficient, and it does not require any involvement of the base station after deployment.

### 7.5 Analysis of the proposed scheme

In this section, we perform the functionality and security analysis of our proposed access control scheme.
7.5 Analysis of the proposed scheme

7.5.1 Computational overhead

Let \( t_{ecm} \), \( t_h \), \( t_{enc} \), and \( t_{dec} \) denote the time taken for performing an elliptic curve scalar multiplication, a one-way hash function \( h(\cdot) \), a symmetric key encryption, and a symmetric key decryption, respectively. In our proposed scheme, during the registration phase the user \( U_j \) and the BS require the computational overhead \( t_h \) and \( t_h \), respectively. During the login phase and the authentication phase, the user \( U_j \), the BS and the sensor node \( SN_i \) require the computational overhead \( 6t_h + t_{dec} + t_{enc} \), \( 3t_h + 2t_{ecm} + 2t_{enc} + t_{eca} \) and \( 3t_h + 3t_{dec} + t_{ecm} \), respectively. As a total, the computational cost becomes \( 14t_h + 8t_{enc} + t_{dec} + 3t_{ecm} + t_{eca} \).

Table 7.5: Size (in bits) of different parameters used for our scheme.

<table>
<thead>
<tr>
<th>Type</th>
<th>Bitwise size</th>
</tr>
</thead>
<tbody>
<tr>
<td>User identifier, ( U_j )</td>
<td>16</td>
</tr>
<tr>
<td>Bootstrapping time, ( T_i )</td>
<td>32</td>
</tr>
<tr>
<td>Node identifier, ( SN_i )</td>
<td>16</td>
</tr>
<tr>
<td>Group identifier, ( G_{idj} )</td>
<td>8</td>
</tr>
<tr>
<td>Access privilege mask, ( APM_j )</td>
<td>64</td>
</tr>
<tr>
<td>Random number, ( RM_{Uj} )</td>
<td>32</td>
</tr>
<tr>
<td>Hash value</td>
<td>160</td>
</tr>
<tr>
<td>Symmetric-key encryption/decryption, ( E_K(M)/D_K(M) )</td>
<td>128</td>
</tr>
</tbody>
</table>

7.5.2 Communication overhead

We consider the communication overhead of our scheme for the login and authentication phase. From the login and authentication phases of our scheme, it is clear that the sensor node \( SN_i \), the BS and the user \( U_j \) need to exchange only four messages. We have calculated the bitwise and packet-wise communication overhead for our proposed scheme during the login and authentication phases. For computing the number of packets required for transmission, we have considered CC2420 transmitter [5]. CC2420 transmitter supports a packet size of 128 bytes, that is, 1024 bits. To calculate the communication overhead, we have used bitwise size of different parameters given in Table 7.5.
In Table 7.6, we have calculated the number of bits and packets required for each message in our scheme during the login and authentication phases. It is noted that we require the communication overhead of 1008 bits and transmission of only 4 packets during the login and authentication phases.

Table 7.6: Message size and number of packets to be transmitted per message for our scheme during the login and authentication phases.

<table>
<thead>
<tr>
<th>Message</th>
<th>Exchange between</th>
<th>Size (in bits)</th>
<th># of packets required</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle U_j</td>
<td></td>
<td>b(R_u</td>
<td></td>
</tr>
<tr>
<td>$\langle E_{UK_j}(SN_i</td>
<td></td>
<td>S_j</td>
<td></td>
</tr>
<tr>
<td>$\langle SN_i</td>
<td></td>
<td>U_j</td>
<td></td>
</tr>
<tr>
<td>$R_{U_j}</td>
<td></td>
<td>T_1</td>
<td></td>
</tr>
<tr>
<td>$G_{id_j}</td>
<td></td>
<td>T_1</td>
<td></td>
</tr>
</tbody>
</table>

### 7.5.3 Storage overhead

During the pre-deployment phase described in Section 7.4.2, prior to deployment a sensor node $SN_i$ mainly requires the storage space due to storing the following node parameters: a unique node identifier $SN_i$, which needs 16 bits; the elliptic curve $E_p(a,b)$, which needs $(160 + 160 + 160) = 480$ bits for storing $p$, $a$ and $b$ each is 160 bits (for security reason, we have considered 160 bits prime $p$ in ECC); the base point $G$, which needs $(160 + 160) = 320$ bits; the secret key $K_i$ with private key $x_i$ for $SN_i$, which need $(160 + 160) = 320$ bits; the base station’s public key $K_{BS}$, which needs $(160 + 160) = 320$ bits; its own master key $MK_{Si}$, which needs 128 bits. Summing up all these terms, the storage space of the sensor node $SN_i$ prior to its deployment becomes 1584 bits.

### 7.5.4 Network scalability

Assume that there will be $m$ cluster heads and $m'$ controller nodes in a hierarchical WBAN (HWBAN) shown in Figure 7.2, where at a hospital ward there are multiple patients. In this figure, a set of sensor nodes are deployed in a patient’s body, which constitutes a WBAN. The WBAN is then associated with a cluster head and a set of
cluster heads are attached to a controller node. For example, if a patient’s body is deployed with 10 regular sensor nodes and there are 1000 patients in various wards to be monitored in the hospital, the total regular sensor nodes is $10 \times 1000 = 10,000$. Again, if 5 cluster heads are attached with a controller node in a ward, we require $1000/5 = 200$ controller nodes in the hospital. As a result, the total nodes to be deployed in HWBAN is 11,200 and these nodes constitute a large-scale network.

In case, when a patient will be monitored at home, the total regular sensor nodes is 10 in a WBAN and only one cluster head is required in that WBAN. In both scenarios, the access control mechanism remains same in our scheme.

![Diagram of hierarchical body area sensor network](image)

Figure 7.2: An example of a hierarchical body area sensor network.

### 7.5.5 Security analysis

In this section, we show that our scheme has the ability to tolerate various known attacks, which are discussed in the following subsections.
Stolen-verifier attack

It is noted that our scheme does not require to store any verifier/password table for password verifications. The insider of the network can not get/steal user’s password because the BS and sensor nodes do not maintain any password/verifier table in order to validate user’s login request. During the registration phase of our scheme, a user \( U_j \) submits his/her identity \( U_j \), masked password \( h(N_j||PW_j) \) securely to the BS. According to our threat model, the BS is considered as a trustworthy entity in the network and it is not compromised by any attacker. Since the secret value \( N_j \) is only known to the user \( U_j \), it is a computationally infeasible problem for the BS to retrieve \( PW_j \) from \( h(N_j||PW_j) \) due to one-way property of the hash function \( h(\cdot) \). Thus, our scheme has the ability to prevent such attack.

Many logged-in users with the same login-id attack

Generally, if the systems which maintain the password table to verify the user login, they are usually vulnerable to this attack. However, in our scheme the BS, and sensor nodes do not maintain any verifier table containing passwords for verification. In the user’s smart card no passwords are also stored. At the time of login, a user \( U_j \) must have a valid smart card with the valid input tuple \( \langle U_j||PW_j||N_j \rangle \). Note that our scheme requires on-card computation for password verification and for login to the WSN, and once the smart card is removed from the system, the login process is aborted. If two users \( U_i \) and \( U_j \) have the same password, due to random secret numbers \( N_i \) and \( N_j \) used in computation of their masked passwords, they will have different the masked passwords. As a result, even if two users have same password, problem of many logged in users with same login id do not arise in our scheme. Thus, our scheme resists the many logged-in users with the same login-id attack.

Resilience against node capture attack

We evaluate the ability of our scheme to tolerate compromised nodes in the network. Let \( P_e(c) \) denote the probability that an adversary compromises a fraction of total secure communications by capturing \( c \) number of sensor nodes in the network. If \( P_e(c) = 0 \), we call our user access control scheme as unconditionally secure against node capture attack. If an attacker captures a sensor node, he/she is able to know the master key along with other information from its memory, because the sensor
nodes are not equipped with tamper-resistant hardware. Note that, each node is given a unique randomly generated master key prior to its deployment and each sensor node establishes a distinct secret session key with a user. Thus, the attacker can only respond with false data to a legitimate user by capturing a sensor node from which the user wants to access data. On the other hand, other non-captured sensor nodes can still communicate with 100% secrecy with the actual real-time data to the legitimate users. As a result, the compromise of a sensor node does not lead to compromise of any other secure communication between the user and the non-captured sensor node in the network and hence, our scheme provides unconditional security against node capture attack.

**Remark 7.1:** Note that in our scheme the session key between the user and a sensor node in BAN is secured after the successful authentication process. This key is used between the sensor and the user to secure the communication channel for the real-time data transmission. However, when a sensor node is physically captured by an attacker from a patient’s body (WBAN), the attacker is able to know the master key along with other information from its memory including the established session key. As in our threat model discussed in Section 7.2, the compromised (captured) nodes can be detected and as a result, the base station (medical server), users and sensor nodes know the ids of the compromised nodes. Consequently, the base station (medical server) alerts the users with the compromised sensor nodes in the network. Hence, another new sensor needs to be deployed in place of captured sensor. In this case, with the help of dynamic node addition phase described in Section 7.4.2 the deployed new sensor will be able to establish new session key shared with the user after successful authentication process.

**Masquerade attack**

In our scheme, an illegal user can not be successful to fabricate the fake login request message to cheat the BS to convince that it is a legal login request in the login phase. At the time of login, the user needs to insert his/her smart card in a card reader and then to provide his/her user id $U_j$, secret value $N_j$, password $PW_j$ and access group id $G_{id_j}$. The smart card then computes the masked password $RPW_j' = h(N_j||PW_j)$ and the hash value $R_{U_j}' = h(RPW_j'||G_{id_j}||RM_{u_j})$ for the user $U_j$ using the stored values of $G_{id_j}$, $RM_{u_j}$ in the smart card. The smart card checks
whether $R'_{U_j} = R_{U_j}$. If this verification passes, the user $U_j$ sends the login request message $\langle U_j || h(R_{U_j} || T_1) || T_1 \rangle$ to the BS. In order to convince the BS that this is a legal remote login request, the illegal user has to know the value of $N_j$ as well as $PW_j$, $N_j$, $G_{id_j}$, and $RM_{u_j}$. As a result, the attacker does not have then any ability to create a fake login request message on behalf of the original user $U_j$. Thus, our scheme resists such kind of attack.

Replay attack

In this attack, an attacker may try to prove as a valid user logging to the BS by sending messages which were previously transmitted by a legal user. However, our scheme makes use of the current system timestamp during the login and authentication phases. Comparison of previous timestamp with current timestamp of the receiver system withstands the replay attacks, since the expected time interval for the transmission delay is very short. Moreover, in the login phase the user sends the message $\langle U_j || h(R_{U_j} || T_1) || T_1 \rangle$ to the BS. Since the attacker can not change the hash value $h(R_{U_j} || T_1)$, the attacker can not also change the value of $T_1$. Thus, an attacker does not have ability to successfully replay previously used messages during the login and authentication phases. As a result, our scheme resists the replay attack.

Privileged-insider attack

Note that during the registration phase of our proposed scheme, the user $U_j$ does not send his/her password $PW_j$ in plaintext. The user $U_j$ sends the masked password $RPW_j = h(N_j || PW_j)$ to the BS. Without knowing the secret value $N_j$, which is only known to the user $U_j$, it is a computationally infeasible problem to retrieve $PW_j$ from $RPW_j$ due to one-way property of the hash function $h(\cdot)$. A privileged-insider of the BS does not have any ability to know the password $PW_j$ of the user $U_j$, and he/she is then unable to impersonate $U_j$ by accessing other servers where $U_j$ could be also a registered user in case $U_j$ may use the same password $PW_j$ for his/her convenience. Thus, our scheme protects such attack.
7.5 Analysis of the proposed scheme

Smart card breach attack

An in [68], although the smart card is assumed safe and cannot be cracked, however there is a risk of smart card crack. If an attacker/intruder attains a smart card and cracks it, we assume that he/she can obtain its stored information, such as $U_j$, $RM_{u_j}$, $h(\cdot)$, $RPW_j$, $G_{id_j}$, $R_U$. However, the attacker has no feasible way to know the user $U_j$’s password $PW_j$ from $RPW_j$ due to one-way property of the hash function $h(\cdot)$. Moreover, from the hash value $R_{U_j} = h(RPW_j||G_{id_j}||RM_{u_j})$, it also a difficult task to know $PW_j$ for $U_j$ due to one-way property of the hash function $h(\cdot)$. Hence, the attacker needs to guess the user $U_j$’s correct password $PW_j$ and secret number $N_j$ in order to pass the password verification during the login phase. In addition, the computation of $N_j$ in the login phase becomes infeasible problem due to one-way property of the hash function $h(\cdot)$. As a result, our scheme prevents smart card breach attack.

Denial-of-service attack

In our scheme, after deployment the sensor node first sends a message to the BS to inform its own bootstrapping time. At the time of authentication, the BS sends the authentication request message to a specific sensor node $SN_i$ from which the user $U_j$ wants to access real-time data inside WBAN. After receiving the request message from the user $U_j$, the sensor node $SN_i$ sends an acknowledgment to the user after successful authentication. If an attacker blocks the messages from reaching the BS and sensor nodes, the BS and sensor node will know about malicious dropping of such control messages [58]. Hence, the denial-of-service attack is not possible in our scheme because at the end of user authentication, an acknowledgment is sent to the user $U_j$.

Formal security analysis of the proposed scheme

In this section, we show through the formal security analysis that our scheme is secure against deriving the user’s password and the base station’s private key by an attacker. We follow the similar proof as in [20], [21], [42], [53], [112], [117], [118] for the formal security analysis of our proposed scheme. We have used the method of contradiction proof [25] for our formal security analysis. Note that one can also prove the formal security in the standard model. However, in this thesis,
we have performed the formal security analysis under the generic group model of cryptography.

We define the following three random oracles for the attacker (adversary), say \( A \):

*Reveal1*: This unconditionally outputs \( k \) from given points \( P \) and \( Q = kP \) in an elliptic curve \( E_p(a,b) \).

*Reveal2*: This unconditionally outputs the plaintext message \( M \) using symmetric-key cryptosystem \( \Omega \) with the help of the relevant public parameters and ciphertext message \( E_{key}(M) \), without knowing the symmetric key, \( key \).

*Reveal3*: This unconditionally outputs the input \( x \) from the corresponding hash value \( y = h(x) \).

**Theorem 7.1.** Let the used symmetric encryption scheme \( \Omega \) be IND-CPA. Then our scheme is secure against deriving a user’s password by an attacker under the assumption that the one-way hash function \( h(\cdot) \) closely behaves like a random oracle.

**Proof.** We need to construct an adversary \( A \) which can derive correctly the user \( U_j \)’s password \( PW_j \). For this purpose, the adversary \( A \) runs the experiment given in Algorithm 6, \( Exp_{UACS,A}^{\text{HASH,IND-CPA}} \), for our proposed user access control scheme \( UACS \).

We define the success probability for \( Exp_{UACS,A}^{\text{HASH,IND-CPA}} \) provided in Algorithm 6 as

\[
Succ_{UACS,A}^{\text{HASH,IND-CPA}} = |2Pr[Exp_{UACS,A}^{\text{HASH,IND-CPA}} = 1] - 1|.
\]

The advantage function for this experiment is given by

\[
Adv_{UACS,A}^{\text{HASH,IND-CPA}}(t_1,q_{R2},q_{R3}) = \max_A \{Succ_{UACS,A}^{\text{HASH,IND-CPA}}\},
\]

where the maximum is taken over all \( A \) with the execution time \( t_1 \), and the number of queries \( q_{R2} \) made to the \( \text{Reveal2} \) oracle and the number of queries \( q_{R3} \) made to the \( \text{Reveal3} \) oracle. Our scheme is provably secure against an adversary \( A \) for deriving a user’s password by an attacker, if

\[
Adv_{UACS,A}^{\text{HASH,IND-CPA}}(t_1,q_{R2},q_{R3}) \leq \epsilon,
\]
Algorithm 6 $\text{Exp}^{\text{HASH,IND−CPA}}_{\text{UACS},\mathcal{A}}$

1: Eavesdrop the message $\langle U_j||h(R_{U_j}||T_1)||T_1 \rangle$ during the login phase, which is sent from the user $U_j$ to the BS.
2: Call $\text{Reveal3}$ oracle on the input $h(R_{U_j}||T_1)$ to retrieve the information $R_{U_j}$ and $T_1$. Let $(R'_{U_j}, T'_1) \leftarrow \text{Reveal3}(h(R_{U_j}||T_1))$.
3: Check if $T'_1$ matches with $T_1$ in the eavesdropped message. If so, call $\text{Reveal3}$ oracle on the input $R'_{U_j} = h(RPW_j||G_{id_j}||RM_{U_j})$, in order to retrieve the information $RPW_j$, $G_{id_j}$ and $RM_{U_j}$. Let $(RPW'_j, G'_{id_j}, RM'_j) \leftarrow \text{Reveal3}(R'_{U_j})$.
4: Call $\text{Reveal3}$ oracle on the input $RPW'_j$ to derive $N_j$ and $PW_j$ of the user $U_j$. Let $(N'_j, PW'_j) \leftarrow \text{Reveal3}(RPW'_j)$.
5: Eavesdrop the message $\langle SN_i||U_j||E_{MK_{SN_i}}(SN_i||U_j||(APM_j \oplus G_{id_j})||R_{U_j}||T_1||T_2)||E_{K_i}(SN_i||U_j||G_{id_j}||T_1) \rangle$ during the login message, which is sent from the BS to a sensor node $SN_i$.
6: Call $\text{Reveal2}$ oracle on the input $E_{K_i}(SN_i||U_j||G_{id_j}||T_1)$.
Let $(SN''_i, U''_j, G''_{id_j}, T'_1) \leftarrow \text{Reveal2}(E_{K_i}(SN_i||U_j||G_{id_j}||T_1))$.
7: if $(G''_{id_j} = G_{id_j})$ and $(T'_1 = T_1)$ then
8: Accept the derived password $PW'_j$ as the correct password $PW_j$ of the user $U_j$.
9: return 1 (Success)
10: else
11: return 0 (Failure)
12: end if

for any sufficiently small $\epsilon > 0$.

Finally, consider the experiment $\text{Exp}^{\text{HASH,IND−CPA}}_{\text{UACS},\mathcal{A}}$. According to this experiment, if the adversary $\mathcal{A}$ can derive correctly the private key of the BS, he/she can win the game. However, it is a computationally infeasible problem due to difficulty of solving the one-way hash function and the indistinguishability of encryption and chosen plaintext attack (IND-CPA). As a result, $\text{Adv}^{\text{HASH,IND−CPA}}_{\text{UACS},\mathcal{A}}(t_1, q_{R_2}, q_{R_3}) \leq \epsilon$, for any sufficiently small $\epsilon > 0$, since it is dependent on $\text{Adv}^{\text{ind−cpa}}_{\Omega,\text{ME}}(t)$ (see Definition 4.1 of Chapter 4) and hardness of inverting the one-way hash function, that is, $\text{Adv}^{\text{HASH}}_{\mathcal{A}}(t)$ (see Definition 2.1 of Chapter 2). Hence, our scheme is provably secure against deriving a user’s password by an attacker.
Theorem 7.2. Let the used symmetric encryption scheme \( \Omega \) be IND-CPA. Under the ECDLP assumption, our scheme is secure against deriving the base station’s private key by an attacker, if the hash function \( h(\cdot) \) closely behaves like a random oracle.

Proof. We need to construct an adversary \( \mathcal{A} \) which can derive correctly the base station \( BS \)’s private key \( x \). For this purpose, the adversary \( \mathcal{A} \) runs the experiment \( \text{Exp}_{2,\text{HASH},\text{IND}-\text{CPA},\text{ECDLP},\text{UACS},\mathcal{A}} \) given in Algorithm 7 for our proposed user access control scheme \( \text{UACS} \).

Algorithm 7 \( \text{Exp}_{2,\text{HASH},\text{IND}-\text{CPA},\text{ECDLP},\text{UACS},\mathcal{A}} \)

1: Eavesdrop the message \( \langle U_j || h(R_{U_j}||T_1)||T_1 \rangle \) during the login phase, which is sent from the user \( U_j \) to the BS.
2: Call \( \text{Reveal}_3 \) oracle on the input \( h(R_{U_j}||T_1) \) to retrieve the information \( R_{U_j} \) and \( T_1 \). Let \( (R_{U_j}',T_1') \leftarrow \text{Reveal}_3(h(R_{U_j}||T_1)) \).
3: Check if \( T_1' \) matches with \( T_1 \) in the eavesdropped message. If so, eavesdrop the message \( \langle E_{UK_j}(SN_i||S_j||Z_j||K_{U_j})||T_2||T_1 \rangle \) during the login phase, which is sent from the BS to the user \( U_j \).
4: Call \( \text{Reveal}_2 \) oracle on the input \( E_{UK_j}(SN_i||S_j||Z_j||K_{U_j}) \). Let \( (SN_i',S_j',Z_j',K_{U_j}') \leftarrow \text{Reveal}_2(E_{UK_j}(SN_i||S_j||Z_j||K_{U_j})) \).
5: Compute \( UK_j' = h(R_{U_j}'||U_j'||T_1'||T_2) \), and encrypt the information using the key \( UK_j' \) as \( E_{UK_j'}(SN_i'||S_j'||Z_j'||K_{U_j}') \). If this encrypted value \( E_{UK_j'}(SN_i'||S_j'||Z_j'||K_{U_j}') \) matches with received \( E_{UK_j}(SN_i||S_j||Z_j||K_{U_j}) \), accept \( K_{U_j}' \) as the correct \( K_{U_j} \).
6: Call \( \text{Reveal}_3 \) oracle on the input \( K_{U_j}' \) to retrieve \( KB_S \). Let \( (SN_i',U_j',K_{BS}',K_i') \leftarrow \text{Reveal}_3(K_{U_j}') \), where \( K_{U_j} = h(SN_i||U_j||K_{BS}||K_i) \).
7: Call \( \text{Reveal}_1 \) oracle on the input \( K_{BS}' \) to derive the private key \( x \) of the BS. Let \( x' \leftarrow \text{Reveal}_1(K_{BS}') \). Compute \( Z_j'' = K_{BS}' + K_i'R_{U_j} \) (mod \( p \)).
8: if \( (Z_j'' = Z_j') \) then
9: Accept the derived \( x' \) as the correct private key \( x \) of the BS.
10: return 1 (Success)
11: else
12: return 0 (Failure)
13: end if
7.6 Formal security verification of our scheme using AVISPA tool

We define the success probability for the experiment in Algorithm 7 as

\[ \text{Succ}_{\text{UACS},A}^{\text{HASH,IND-CPA,ECDLP}} = |3\Pr[\text{Exp}_{\text{UACS},A}^{\text{HASH,IND-CPA,ECDLP}} = 1] - 1|. \]

The advantage function for this experiment is given by

\[ \text{Adv}_{\text{UACS},A}^{\text{HASH,IND-CPA,ECDLP}}(t_2, q_{R_1}, q_{R_2}, q_{R_3}) = \max_A \{ \text{Succ}_{\text{UACS},A}^{\text{HASH,IND-CPA,ECDLP}} \}, \]

where the maximum is taken over all \( A \) with the execution time \( t_2 \), and the number of queries \( q_{R_1}, q_{R_2}, q_{R_3} \) made to the \( \text{Reveal}1 \) oracle, \( \text{Reveal}2 \) oracle, and \( \text{Reveal}3 \) oracle, respectively. Our scheme is called provably secure against an adversary \( A \) for deriving the base station’s private key by an attacker, if

\[ \text{Adv}_{\text{UACS},A}^{\text{HASH,IND-CPA,ECDLP}}(t_2, q_{R_1}, q_{R_2}, q_{R_3}) \leq \epsilon, \]

for any sufficiently small \( \epsilon > 0 \).

Consider the experiment \( \text{Exp}_{\text{UACS},A}^{\text{HASH,IND-CPA,ECDLP}} \). According to the experiment, if the adversary \( A \) can derive correctly the user password, he/she can win the game. However, it is a computationally infeasible problem due to difficulty of solving the one-way hash function, the indistinguishability of encryption and chosen plaintext attack (IND-CPA) and elliptic curve discrete logarithm problem (ECDLP). As a result, \( \text{Adv}_{\text{UACS},A}^{\text{HASH,IND-CPA,ECDLP}}(t_2, q_{R_1}, q_{R_2}, q_{R_3}) \leq \epsilon \), for any sufficiently small \( \epsilon > 0 \), since it is dependent on \( \text{Adv}_{\text{UASC},\text{ME}}^{\text{ind-CPA}}(l) \) (see Definition 4.1 of Chapter 4), \( \text{Adv}_{D,E_p(a,b)}^{\text{ECDLP}}(t) \) (see Definition 2.3 of Chapter 2) and \( \text{Adv}_{A}^{\text{HASH}}(t) \) (see Definition 2.1 of Chapter 2). Hence, our scheme is provably secure against deriving the private key of the BS by an attacker.

\[ \square \]

7.6 Formal security verification of our scheme using AVISPA tool

In this section, we simulate our scheme for the formal security verification. Through the simulation results using the widely-accepted AVISPA tool we show that our scheme is secure against passive and active attacks including the replay and man-in-the-middle attacks. For this purpose, we first implement our scheme in the HLPSL language and then simulate the implemented protocol to show that our scheme is secure.
role bob (U, BS, SN : agent,
   MKsi : symmetric_key,
   MKuj: symmetric_key,
   H : hash_func,
   Snd, Rcv: channel(dy))
played_by U
def=
local State : nat,
Uj, RPWj, APMj, RMuj, Nj, PWj, UKj : text,
Ruj, Kuj, SNi, Sj, Zj, Ki, Kbs, Gldj, RNui : text,
T1, T2 : text
const alice_server, server_bob, bob_server, bob_alice,
subs1, subs2, subs3, subs4, subs5, subs6 : protocol_id
init State := 0
transition
1. State = 0 \& Rcv(start) =>
   State' := 1 \&
   RPWj' := H(PWj.Nj)
   \& RMuj' := new()
   \& Snd(U.BS.{Uj.RPWj'.GIdj.RMuj'}.MKuj)
   %smart card values
   State' := 2 \&
   secret({Ki}, subs1, {SN,BS})
   \& secret({MKsi}, subs2, {SN,BS})
   \& secret({RMuj'}, subs3, {U,BS})
   \& secret({Kbs}, subs4, {SN,BS})
   \& secret({APMj, Gldj}, subs5, {U,BS})
   \& secret({PWj, Nj}, subs6, U)
   \& T1' := new()
   \& Snd(U.BS.Uj.H(H(H(PWj.Nj).Gldj.RMuj').T1').T1')
   \& witness(U, BS, bob_server, T1')
   State' := 3 \&
   UKj' := H(Ruj.Uj.T1'.T2')
   H(T1'.Sj.Zj))
   \& witness(U, SN, bob_alice, T1')
end role

Figure 7.3: Role specification in HLPSL for the user $U_j$ of our scheme.

7.6.1 Specifying our scheme

We have implemented our scheme in HLPSL language. In this implementation, we have three basic roles: alice, server and bob which represent the participants: the sensor node $SN_i$, the BS and the user $U_j$, respectively. We have also defined the session and environment in our scheme.

Figure 7.3 illustrates the role specification for the user $U_j$ in HLPSL. During the registration phase $U_j$ sends the message $\langle U_j||RPW_j||G_{sdj}||RM_{u_j} \rangle$ securely to the BS with the $Snd( )$ operation. The type declaration $channel(dy)$ indicates that
the channel for the Dolev-Yao threat model (as described in our threat model in Section 7.2). \( U_j \) then waits for the smart card containing the information in the message \( \langle U_j \| RM_{a_j} \| h(\cdot) \| RPW_j \| G_{id_j} \| R_{U_j} \rangle \) securely from the BS from the \( Rcv(\cdot) \) operation. The intruder will have the ability to intercept, analyze, and/or modify messages transmitted over the insecure channel. During the login phase, \( U_j \) sends the login request message \( \langle U_j \| h(R_{U_j} \| T_1) \| T_1 \rangle \) to the BS. In reply, the BS sends the messages \( \langle E_{U_{K_j}}(SN_i \| S_j \| Z_j \| K_{U_j}) \| T_2 \| T_1 \rangle \) to \( U_j \). Finally, during the authentication phase \( U_j \) sends the authentication request message \( \langle SN_i \| U_j \| Z_j \| S_j \| E_{K_{U_j}}(SN_i \| U_j \| R_{U_j} \| G_{id_j} \| T_1 \| S_j \| Z_j) \| h(T_1 \| S_j \| Z_j) \rangle \) to the sensor node \( SN_i \).

Figure 7.4 shows the role specification for the BS in HLPSL language. During the post-deployment phase, the BS receives the message \( \langle SN_i \| T_i \| E_{MK_{S_i}}(K_i \| SN_i \| T_i) \rangle \) from the sensor node \( SN_i \). During the registration phase after receiving the message \( \langle U_j \| RPW_j \| G_{id_j} \| RM_{a_j} \rangle \) securely from the user \( U_j \), the BS sends securely sends the smart card containing the information in the message \( \langle U_j \| RM_{a_j} \| h(\cdot) \| RPW_j \| G_{id_j} \| R_{U_j} \rangle \) to the user \( U_j \). In the login phase, when the BS receives the message \( \langle U_j \| h(R_{U_j} \| T_1) \| T_1 \rangle \) from the user \( U_j \), the BS sends the messages \( \langle E_{U_{K_j}}(SN_i \| S_j \| Z_j \| K_{U_j}) \| T_2 \| T_1 \rangle \) and \( \langle SN_i \| U_j \| E_{MK_{S_i}}(SN_i \| U_j \| (APM_j \oplus G_{id_j}) \| R_{U_j} \| T_1 \| T_2) \| E_{K_i}(SN_i \| U_j \| G_{id_j} \| T_1) \rangle \) to \( U_j \) and \( SN_i \). Finally, during the authentication phase, the BS sends the message \( \langle SN_i \| U_j \| Z_j \| S_j \| E_{K_{U_j}}(SN_i \| U_j \| R_{U_j} \| G_{id_j} \| T_1 \| S_j \| Z_j) \| h(T_1 \| S_j \| Z_j) \rangle \) to the sensor node \( SN_i \).

In Figure 7.5, we have implemented the role specification for the sensor node \( SN_i \) in HLPSL language. In the the post-deployment phase, the sensor node \( SN_i \) sends the message \( \langle SN_i \| T_i \| E_{MK_{S_i}}(K_i \| SN_i \| T_i) \rangle \) to the BS. In the login phase, the sensor node \( SN_i \) receives the message \( \langle SN_i \| U_j \| E_{MK_{S_i}}(SN_i \| U_j \| (APM_j \oplus G_{id_j}) \| R_{U_j} \| T_1 \| T_2) \| E_{K_i}(SN_i \| U_j \| G_{id_j} \| T_1) \rangle \) from the BS. During the authentication phase, the sensor node receives the authentication request message \( \langle SN_i \| U_j \| Z_j \| S_j \| E_{K_{U_j}}(SN_i \| U_j \| R_{U_j} \| G_{id_j} \| T_1 \| S_j \| Z_j) \| h(T_1 \| S_j \| Z_j) \rangle \) from the user \( U_j \).

witness(A,B,id,E) declares for a (weak) authentication property of \( A \) by \( B \) on \( E \), declares that agent \( A \) is witness for the information \( E \); this goal will be identified by the constant \( id \) in the goal section. request(B,A,id,E) means for a strong authentication property of \( A \) by \( B \) on \( E \), declares that agent \( B \) requests a check of the value \( E \); this goal will be identified by the constant \( id \) in the goal section. The intruder is always denoted by \( i \).

Finally, the specifications in HLPSL language for the role of session, goal and environment are specified in Figures 7.6 and 7.7. In the session segment, all the basic roles: alice, server and bob are instanced with concrete arguments. The top-level
Figure 7.4: Role specification in HLPSL for the BS of our scheme.

role (environment) is always defined in the specification of HLPSL language. This role contains the global constants and a composition of one or more sessions, where the intruder may play some roles as legitimate users. The intruder also participates in the execution of protocol as a concrete session. In our scheme, six secrecy goals and four authentications are verified.
7.6 Formal security verification of our scheme using AVISPA tool

7.6.2 Analysis of results

We have simulated our scheme for OFMC and CL-AtSe beck-ends using the AVISPA web tool [4]. The simulation results are shown in Figures 7.8 and 7.9. The summary of the results are as follows:

- OFMC reports the protocol is safe.
- CL-AtSe reports the protocol is safe.

Thus, it is clear that our scheme is secure against passive and active attacks including the replay and man-in-the-middle attacks.
role session(SN, BS, U : agent, 
  % H is hash function 
  MKsi : symmetric_key, 
  MKuj : symmetric_key, 
  H : hash_func )
def=
local US, UR, SS, SR, VS, VR: channel(dy)
composition
  alice(SN, BS, U, MKsi, H, US, UR)
\ server(BS, U, SN, MKsi, MKuj, H, SS, SR)
\ bob(U, BS, SN, MKsi, MKuj, H, VS, VR)
end role

Figure 7.6: Role specification in HLPSL for the session of our scheme.

role environment()
def=
const sn, bs, u : agent,
      mksi : symmetric_key,
      mkuj : symmetric_key, 
      h : ... bob_server
authentication_on bob_alice
authentication_on alice_bob
end goal
environment()

Figure 7.7: Role specification in HLPSL for the goal and environment of our scheme.

7.7 Performance comparison with other related schemes

In this section, we compare the performance of our scheme with the relevant existing access control schemes, such as Mahmud et al.’s scheme [109], Wang et al.’s scheme
7.7 Performance comparison with other related schemes

Figure 7.8: The result of the analysis using OFMC of our scheme.

Figure 7.9: The result of the analysis using CL-AtSe of our scheme.

[138] and Le et al.’s scheme [93].
Table 7.7: Time complexity of various operations in terms of $t_{mul}$.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{ecm}$</td>
<td>$\approx 1200 t_{mul}$</td>
</tr>
<tr>
<td>$t_{sigr}$</td>
<td>$\approx 2405.36 t_{mul}$</td>
</tr>
<tr>
<td>$t_i$</td>
<td>$\approx 3 T_{mul}$</td>
</tr>
<tr>
<td>$t_{add}$</td>
<td>Negligible</td>
</tr>
<tr>
<td>$t_h$</td>
<td>$\approx 0.36 t_{mul}$</td>
</tr>
<tr>
<td>$t_{enc}$</td>
<td>$\approx 0.15 t_{mul}$</td>
</tr>
<tr>
<td>$t_{dec}$</td>
<td>$\approx 0.15 t_{mul}$</td>
</tr>
<tr>
<td>$t_{mac}$</td>
<td>$\approx t_h$</td>
</tr>
<tr>
<td>$t_{sigr}$</td>
<td>$\approx 1204.36 t_{mul}$</td>
</tr>
</tbody>
</table>

### 7.7.1 Comparison of computational costs

We have used the notations for comparison of computational cost between our scheme and other schemes as follows. $t_{ecm}$, $t_{eca}$, $t_i$, $t_{add}$, $t_{mul}$, $t_h$, $t_{enc}$, $t_{dec}$, $t_{ecenc}$, $t_{ecdec}$, $t_{mac}$, $t_{sigr}$, and $t_{sigr}$ denote the time taken for performing one ECC point multiplication over finite field $GF(2^{163})$, ECC point addition over finite field $GF(2^{163})$, modular inverse over finite field $GF(2^{163})$, modular addition over finite field $GF(2^{163})$, modular multiplication over finite field $GF(2^{163})$, hashing operation $h(\cdot)$, AES encryption, AES decryption, ECC encryption over finite field $GF(2^{163})$, ECC decryption over finite field $GF(2^{163})$, MAC operation, ECC signature generation over finite field $GF(2^{163})$, and ECC signature verification over finite field $GF(2^{163})$, respectively. We have considered the time taken for one MAC operation as that for one hashing operation for simplicity. The quantitative analysis of [89] shows that the computation of a point multiplication requires approximately 1200 field multiplications; an elliptic curve point addition requires one field inversion and two field multiplications; the computation of a field inversion requires approximately three field multiplications; the computation of elliptic curve encryption and decryption require approximately 2405 and 1205 field multiplications respectively [59], [125]; and the cost of field addition is negligible. Further, a 1024-bit modular multiplication takes 41 times longer than a field multiplication in finite field $GF(2^{163})$. The results of Wong et al. [143] show the speed for AES encryption and decryption, hash function using SHA-1 and 1024-bit modular multiplication. In Table 7.7, the time complexity of various operations in terms of $t_{mul}$ are listed according to the analysis results reported in [147].

We have compared the computational complexity using both formulated results and rough quantitative analysis in Table 7.8 for different phases: the registration, login and authentication phases of [93], [109], [138], and our scheme. It is clear that
Table 7.8: Comparison of computational costs for different phases in different access control schemes.

<table>
<thead>
<tr>
<th>Phase</th>
<th>User/Node</th>
<th>[93]</th>
<th>[138]</th>
<th>[109]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>$U_j$</td>
<td>$2t_{ecm} + t_{siggen}$</td>
<td>$t_h + 3t_{ecm}$</td>
<td>$t_h$</td>
<td>$t_h$</td>
</tr>
<tr>
<td></td>
<td>BS</td>
<td>$t_{mac}$</td>
<td>$t_{siggen}$</td>
<td>$t_{dec}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$SN_i$</td>
<td>$t_{mac} + t_h$</td>
<td>$t_{siggen} + 3t_{ecm}$</td>
<td>$3t_h + 2t_{ecm}$</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>$U_j$</td>
<td>$t_h + t_{sigver}$</td>
<td>$t_{ecm} + 2t_{mac}$</td>
<td>$6t_h + t_{dec}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+ A</td>
<td>$2t_{sigver} + 2t_{mac}$</td>
<td>$t_{siggen}$</td>
<td>$+t_{enc}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BS</td>
<td>$3t_{mac} + t_h$</td>
<td>$t_{mac} + 2t_{siggen}$</td>
<td>$3t_h + t_{enc} + t_{ecm}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$SN_i$</td>
<td>$t_{mac} + t_h$</td>
<td>$t_{sigver}$</td>
<td>$+3t_{dec}$</td>
<td></td>
</tr>
<tr>
<td>Total cost</td>
<td></td>
<td>$4t_h + 2t_{ecm}$</td>
<td>$2t_h + 7t_{ecm}$</td>
<td>$14t_h + 8t_{enc}/t_{dec}$</td>
<td></td>
</tr>
<tr>
<td>Rough estimation</td>
<td></td>
<td>$1202t_{mul}$</td>
<td>$8413t_{mul}$</td>
<td>$3611t_{mul}$</td>
<td></td>
</tr>
</tbody>
</table>

Notes: R: Registration; L: Login; A: Authentication.

compared with the other existing schemes, the computational cost of our scheme is much less. Thus, our scheme is much suitable for the resource-constrained sensor nodes.

7.7.2 Comparison of communication costs

In Table 7.9, we have compared the communication costs among our scheme and the other related schemes [93], [109] and [138] in terms of the total number of bits and the total number of packets required for transmissions during all phases. From this table, it is noted that our scheme requires six message exchanges and among them where sensor node is directly involved requires only one message transmission compared to that for the other schemes where sensor node is directly involved. As a result, our scheme is significantly efficient in term of communication cost as
comparing to the other related schemes.

Table 7.9: Comparison of communication costs between the proposed scheme and the other schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$I_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Le et al. [93]</td>
<td>2208</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Mahmud-Morogan [109]</td>
<td>1132</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Wang et al. [138]</td>
<td>2544</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Ours</td>
<td>1400</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

Notes: $I_1$: Total number of bits transmission required for messages of all phases for the schemes; $I_2$: Total number of packets transmissions during all phases for the schemes; $I_3$: Total number of message transmissions during all phases for the schemes.

We have further calculated the total number of bits required for all the messages during all phases for the access control schemes. We have calculated the number of packets required for transmission of a message for the CC2420 transceiver [5] which supports a packet of size 128 bytes, that is, 1024 bits. The results shown in Table 7.9 demonstrates that our scheme is also efficient compared to other related schemes.

### 7.7.3 Comparison of energy costs

Since sensor nodes are resource-constrained, we have mainly considered the energy cost of a sensor node during the login and authentication phases. We have compared the energy cost of a sensor node during the login and authentication phases between our scheme, Le et al.’s scheme [93], Mahmud-Morogan’s scheme [109] and Wang et al.’s scheme [138] in Table 7.10. As in [21], [57], the energy cost of a sensor node is considered due to both computational and communication costs involved during the login and authentication phases. In wireless communication, energy of sensor nodes mainly goes for transmissions and receptions of messages/packets rather than computing. Since our scheme requires no message or packet transmissions during the login and authentication phases as compared to other schemes, the energy spent by sensor nodes is significantly less as compared to other schemes.
Table 7.10: Comparison of energy cost of a sensor node during the login and authentication phases between our scheme and other schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Sensor node’s energy cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Le et al. [93]</td>
<td>three MAC operations + one hash operation + three message transmissions</td>
</tr>
<tr>
<td>Mahmud-Morogan [109]</td>
<td>one ECC-point addition + three ECC-point multiplication + one hash operation + two MAC operations + three message transmissions</td>
</tr>
<tr>
<td>Wang et al. [138]</td>
<td>two hash operations + one ECC-signature generation + two ECC-signature verifications + two message transmissions</td>
</tr>
<tr>
<td>Ours</td>
<td>three hash operations + one ECC-point multiplication + three symmetric-key decryptions + no message transmissions</td>
</tr>
</tbody>
</table>

7.7.4 Comparison of functionality

In this section, we have compared the functionality of our scheme with other schemes [93], [109] and [138] in Table 7.11. It is noted that Le et al.’s scheme [93] is based on ECC where it supports session key establishment between the user and the sensor node and mutual authentication between the user and the sensor node. Their scheme does not support the user’s password change and dynamic sensor node addition phase after initial deployment. In Wang et al.’s scheme [138], ECC is used as the cryptographic technique. It supports session key establishment between the user and the sensor node and mutual authentication between the user and the sensor node, whereas it does not support the user’s password change and dynamic sensor node addition phase after initial deployment. In Mahmud-Morogan’s scheme [109], identity-based signature approach with ECC is based as the cryptographic technique. As in other schemes, their scheme supports session key establishment between the user and the sensor node and mutual authentication between the user and the sensor node, whereas it does not support the user’s password change and dynamic sensor node addition phase after initial deployment. Finally, we see that our scheme uses the hybrid approach using both ECC and symmetric-key cryptosys-
Table 7.11: Comparison of functionality analysis between the proposed scheme and the other schemes.

<table>
<thead>
<tr>
<th></th>
<th>[93]</th>
<th>[138]</th>
<th>[109]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cryptographic technique</td>
<td>ECC</td>
<td>ECC</td>
<td>IBS with ECC (ECC with symmetric-key cryptosystem)</td>
<td>Hybrid</td>
</tr>
<tr>
<td>Session key establishment</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>User password change</td>
<td>Not available</td>
<td>Not available</td>
<td>Not available</td>
<td>Supported</td>
</tr>
<tr>
<td>Dynamic sensor node addition</td>
<td>Not available</td>
<td>Not available</td>
<td>Not available</td>
<td>Supported</td>
</tr>
<tr>
<td>Mutual authentication between user and sensor node</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

Our scheme supports session key establishment between the user and the sensor node and mutual authentication between the user and the sensor node. In addition, our scheme supports the user’s password change and dynamic sensor node addition phase after initial deployment, which are very important requirements for an ideal user access scheme designed in WSNs. Furthermore, our scheme provides the mutual authentication between the BS and the sensor nodes.

Finally, in Table 7.12, we have compared the results of the formal security verification of all schemes, under AVISPA model checkers. From this table, it is clear that Wang et al.’s scheme [138] and our scheme are secure against passive and active adversaries, while Le et al.’s scheme [93] and Mahmud-Morogan’s scheme [109] are insecure against passive and active attacks including the replay and man-in-the-middle attacks.
Table 7.12: Summary of the results of formal security verification using OFMC and CL-AtSe backends for our scheme and existing schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Results using OFMC and CL-AtSe backends</th>
</tr>
</thead>
<tbody>
<tr>
<td>Le et al. [93]</td>
<td>Unsafe</td>
</tr>
<tr>
<td>Mahmud-Morogan [109]</td>
<td>Unsafe</td>
</tr>
<tr>
<td>Wang et al. [138]</td>
<td>Safe</td>
</tr>
<tr>
<td>Ours</td>
<td>Safe</td>
</tr>
</tbody>
</table>

7.8 Summary

In this chapter, we have proposed a new user access control scheme suitable in wireless body area networks healthcare and patient monitoring applications. The proposed scheme allows the user to authenticate at the sensor node inside WBAN under certain access privilege. After successful authentication, both the user and the sensor node from which user wants to access the real-time data, will be able to establish a secret session key between them. Later using this session key, the user can contact the sensor node for the real-time data inside WBANs. Our scheme provides unconditional security against node capture attack and also prevents other known attacks such as denial-of-service attack, masquerade attack, stolen-verifier attack, many logged-in users with the same login-id attack, replay attack, privileged-insider attack, smart card breach attack, and man-in-the-middle attack. The proposed scheme supports dynamic node addition phase and in that case, there is no need to update stored information in the user’s smart card for accessing real-time data from the added/replaced sensor nodes in the network. Using AVISPA tool, we have shown that our scheme is secure against the passive attacks and the active attacks including the replay attack and the man-in-the-middle attack. Our scheme also supports other features such as changing password by the user freely and locally without contacting the BS at any time security reasons, whereas other existing schemes do not support this feature. Our scheme also supports dynamic sensor node addition after initial deployment whereas other existing approaches do not have this important feature. Our scheme is also efficient in terms of communication, computation, storage and energy overheads. Overall, higher security along with low
communication and computational costs make our scheme much appropriate for practical applications in the emerging healthcare field compared to other existing approaches.
Chapter 8

Conclusion and Future Works

This chapter summarizes the major contributions of the thesis. It also highlights the road-map for future research directions in the field of sensor network security.

8.1 Contributions

The contributions of the thesis are summarized as follows. In this thesis, we have proposed several novel schemes in wireless sensor networks in the following areas:

- User Authentication in hierarchical wireless sensor networks
- Certificate-based access control in distributed wireless sensor networks
- Certificate-less access control in distributed wireless sensor networks
- User access control in hierarchical wireless body area sensor networks

The first contribution provided in Chapter 4 is to design a new dynamic password-based user authentication scheme in the hierarchical wireless sensor networks in order to provide access to the real-time data by authorized users directly at the node level and also making it possible for users to communicate with the nodes in order to have responses to their queries. As most of the applications in wireless sensor network (WSN) are real-time based, so users are generally interested in accessing real-time information. For this proposed scheme, we have considered a hierarchical or heterogeneous wireless sensor network (HWSN), where after successful authentication, both the user and the cluster head will be able to establish
a secret session key between them. In future, the user can communicate with the
cluster head using this established session key for accessing the real-time data inside
WSN. Our proposed scheme achieves better security and efficiency as compared to
those for other existing related password-based approaches in WSN. In addition,
our scheme has merit to change dynamically the user’s password locally without
the help of the base station or gateway node. Furthermore, our scheme supports
dynamic nodes addition after the initial deployment of nodes in the existing sensor
network. Further, we have simulated this proposed scheme for formal security verifi-
cation using the widely-accepted AVISPA tool. AVISPA tool ensures that whether a
protocol is insecure against possible passive and active attacks, including the replay
and man-in-the-middle attacks. Using the AVISPA model checkers, we have shown
that our proposed scheme is secure against possible passive and active attacks.

In our second contribution (Chapter 5), we have proposed a new certificate-
based access control scheme for the large-scale distributed wireless sensor networks,
which not only identifies the identity of each node but it has also ability to dif-
ferentiate between old nodes and new nodes. The proposed scheme does not re-
quire involvement of the base station during authentication and key establishment
processes, and it can be easily implemented as a dynamic access control protocol.
In addition, our scheme significantly reduces communication costs in order to au-
thenticate neighbor nodes among each other and establish symmetric keys between
neighbor nodes as compared with existing approaches. Further, our scheme is secure
against different attacks and unconditionally secure against node capture attacks.
The simulation results of our scheme using the AVISPA tool ensure that our scheme
is safe.

The third contribution (Chapter 6) is the enhancement of Huang’s access con-
trol scheme [78] for the large-scale distributed wireless sensor networks, which is
based on the elliptic curve cryptosystem. We have identified that though Huang’s
scheme [78] is efficient, but it has a fatal weakness such as it is vulnerable to an
active attack known as man-in-the-middle attack. In order to remedy that weak-
ness we propose a more efficient and secure certificate less access control scheme as
compared with Huang’s scheme. Further, our scheme is significantly better in terms
of performance and security compared with other related access control schemes.
In fact, our scheme requires significantly less communication costs as compared to
other related schemes. Moreover, we have simulated our scheme for formal security
analysis using the AVISPA tool and shown that our scheme is secure.

The final contribution (Chapter 7) is devoted to the user access control in hierarchical wireless body area networks (WBANs) used for healthcare and patient monitoring applications. We have proposed a new user access control scheme for WBAN. The proposed scheme makes use of the group-based user access id, access privilege mask as well as password. The elliptic curve cryptography-based public key cryptosystem is used to ensure that any particular legitimate user can access only those information for which he/she is permitted to access them. We have shown that our scheme performs better than the previously existing user access control schemes. Through the security analysis, we have shown that our scheme is secure against possible known attacks. Furthermore, through the formal security verification using the AVISPA tool we have shown that our scheme is also secure against passive and active attacks.

8.2 Future research directions

In this section, we suggest some directions for possible future works. Several research directions are worth investigating as follows.

One of the future research works includes extending our password-based user authentication scheme in Chapter 4 using the biometric features of a user. Recent research [39], [46], [48], [96], [100], [149] demonstrates that the biometric-based user authentication in WSN becomes inherently more reliable and secure than usual traditional password-based user authentication schemes. There are the following major advantages of using biometric keys (for example, fingerprints, faces, irises, hand geometry and palm-prints, etc.) over traditional passwords (as pointed out in [96]):

- Biometric keys can not be lost or forgotten.
- Biometric keys are very difficult to copy or share.
- Biometric keys are extremely hard to forge or distribute.
- Biometric keys can not be guessed easily.
- Someone’s biometrics is not easy to break than others.
A biometric system is considered as an automatic recognition system which operates by acquiring biometric data from an individual, extracting a feature set from the acquired data, and comparing this feature set against the template set in the database. Biometric verification allows one to confirm or establish an individual’s identity. Recently biometric-based user authentication along with passwords in WSNs have drawn considerable attention in research [43], [44], [45], [46], [48], [112], [113], [149]. In future, we are interested to work on designing novel biometric-based user authentication schemes for wireless sensor networks.

Another future research direction includes proposing novel attribute-based user access control schemes in WSNs. In attribute-based user access control, any authorized user should have the access right only to those data for which he/she has an access privilege. Using attribute-based encryption (ABE) and ID-based encryption (IBE), data intended for a user can be encrypted based on attributes or privileges assigned to that user by a trusted authority. Hence, only the users, who have the required attributes, will be able to decrypt the data.
Bibliography


