SECURITY FOR MOBILE HEALTH CARE SYSTEMS

KALVINDER SINGH
BSc(Hons)

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY
SCIENCE, ENVIRONMENT, ENGINEERING AND TECHNOLOGY
GRIFFITH UNIVERSITY

Submitted in fulfilment
of the requirements of the degree of
DOCTOR OF PHILOSOPHY

March 2013
The ageing population and the increase in chronic diseases have placed a considerable financial burden on health care services. Mobile health care systems can play an important role in reducing the costs. The pervasiveness of smart phones and the evolution of Internet-of-Things are increasing the potential for mobile health care systems to remotely manage the health of a patient or the elderly. Smart phones and small devices, such as body sensors, are used to remotely monitor patients suffering from chronic diseases and allow them to have relatively independent lives. A mobile health care system may require a degree of real-time monitoring or data collection. For instance, a medical emergency will require data sent to medical staff as quickly as possible, rather than the data sent after a few hours or days. The problem will be more complex if there is a requirement that commands sent to body sensors need to be in real-time. If the system recognises a possible medical emergency, it may need to notify other devices immediately to start recording data or to actuate (for example, an insulin pump and a defibrillator).

It is important for security protocols to be reliable in mobile health care systems. When sensors are deployed in an open mobile health care environment, they have a likelihood of becoming the target of adversaries. In an open environment, an individual device or an intermediate sensor node may not be completely and permanently trustworthy. This thesis provides a detailed description and limitations of existing, reliable key establishment protocols. To create a new reliable key establishment protocol a survey of key establishment protocols that use symmetric keys is performed. Several features from different protocols are combined to form a new reliable secure protocol as well as new protocols for updating existing keys. An analysis of the security of many of the protocols is supplied. Mobile health care systems may have different types of networks, and hence have different network resources in different areas of the network. New reliable secure protocols are created that take advantage of the different networks in a system. An implementation of most of the protocols is tested in a simulator, emulator and real hardware. An analysis of the performance results from each of the test-beds is provided. A recommendation map is also created based on the network topology, security requirements and performance consideration.
A user-friendly initial key establishment protocol for devices is required by patients and users of a mobile health care system. This thesis specifies a new family of protocols where keys can be established using changing low entropy data, such as physiological signals found in the body. A security analysis is supplied for the physiological data changing at different rates. A performance comparison when using RSA and elliptic curves over the new family of protocols is also supplied. Another set of protocols are created, so that sensors measuring different physiological data can establish a secure connection. Also, a new group key protocol is developed, using physiological data, to secure all the nodes. The implementation of the protocols is tested in a simulator, emulator and real hardware; providing performance results from each of the test-beds.

It is important from a security or information assurance point of view to have a formal methodology to validate the system with the new protocols. A new method is developed to validate security protocol assumptions in a complex system. A mobile health care system is defined, and some of the security protocols that are created are mapped into the home health care system. The server and sensor components as well as some of the new security protocols are defined in the requirement behaviour trees. This thesis demonstrates how security assumptions found in each of the new protocols can be validated in the mobile health care system. Further validation is done on showing the actual entropy values of physiological data. A database of ECG values is used, and a security analysis of the protocols is completed using worst case scenarios. The validation step resulted in a new set of security protocols, where the sensed data with a level of uncertainty or error margin is taken into account. The final validation step is an implementation of sensing the physiological data using pervasive smart phones and low-cost Arduino devices, to confirm that pervasive and low-cost devices can be used in the new protocols.

The work presented in this thesis may be extended in a number of directions. A complete mobile health care system with mobile phones and sensors could be developed to shed light on limitations and enhancements required in the new security protocols. It can also provide a platform for development in strategically important areas in health care. Also, developing a sensor that can accurately measure physiological data is another challenging research direction. The higher the accuracy the more secure the new protocols can become. A database of ECG data should be further examined to include all the data, instead of the subset that is used in this thesis. Investigation about physical layer security where the randomness in the channels could be used in the new protocols is another topic for research.
DECLARATION

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

(Signed) ____________________________
Kalvinder Singh
11th March, 2013
## Contents

1 Introduction .......................... 1  
  1.1 Overview .............................. 1  
  1.2 Research Scope ......................... 4  
  1.3 Thesis Outline .......................... 7  

2 Background ............................ 9  
  2.1 Security for Mobile Health care Systems ................................. 9  
  2.1.1 Mobile Health Care Sensor Systems ................................. 12  
  2.1.2 Compatibility Issues between Different Environments .................. 15  
  2.1.3 Limitations with Power and Security ................................. 15  
  2.1.4 Information Assurance, Security and Privacy Threats .................. 17  
     2.1.4.1 Impersonation of the User ................................... 18  
     2.1.4.2 Impersonation of the Service ................................... 18  
     2.1.4.3 Modification of Software ................................... 19  
     2.1.4.4 Modification of Data ................................... 19  
     2.1.4.5 Disclosure of Sensitive Data ................................... 19  
     2.1.4.6 Denial of Service ................................... 19  
     2.1.4.7 Repudiation ................................... 20  
     2.1.4.8 Difficulty in using Complex Technology ......................... 20  
     2.1.4.9 Inability to Keep Track of Changing Technology ................. 20  
     2.1.4.10 Lack of Trust on the System ................................... 20  
     2.1.4.11 Expectation of Reliability ................................... 20  
     2.1.4.12 Expectation of Real-Time Communication ......................... 21  
  2.1.5 Countermeasures to the Threats ................................... 21  
     2.1.5.1 Privacy ................................... 22  
     2.1.5.2 Mapping the countermeasures ................................... 22  
     2.1.5.3 Cryptography ................................... 24  
     2.1.5.4 Key Management ................................... 24  
     2.1.5.5 Pair-wise Key Establishment ................................... 25  
     2.1.5.6 Random Key Establishment ................................... 25
2.1.5.7 Key Distribution Centre Schemes .................................... 26
2.1.5.8 Environment Information ........................................... 27
2.1.5.9 Reliable Security ..................................................... 28
2.1.5.10 Group Keys ......................................................... 29
2.1.6 Smart Cards and RFIDs in Mobile Health Care .................. 30
2.2 Design and Verification of Complex Systems ....................... 32
  2.2.1 Formal Verification .................................................. 32
  2.2.2 Use of GDM in Complex Systems ................................. 34
2.3 Measuring Physiological Data .......................................... 36
  2.3.1 Electrocardiography ................................................ 36
  2.3.2 Photoplethysmographic ............................................ 38
2.4 Sensor Development Platforms ...................................... 39
  2.4.1 Sensor Motes ....................................................... 39
    2.4.1.1 Body Sensor Networks ...................................... 41
  2.4.2 Arduino Devices ................................................... 42
  2.4.3 Android Smart Phones ............................................ 42
2.5 Limitations ............................................................. 44
  2.5.1 Reliability of Security Protocols ................................ 44
  2.5.2 Physiological Data as a Key ..................................... 45
  2.5.3 Validation of Security Assumptions ............................. 45
  2.5.4 Validation of Inter-Pulse Interval ............................... 46
2.6 Conclusion ............................................................... 47

3 Methodology .............................................................. 49
  3.1 Research Approach ................................................... 49
  3.2 Research Questions ................................................ 50
  3.3 Hypothesis ............................................................. 51
  3.4 Research Methodology ............................................... 52
    3.4.1 Investigate ....................................................... 54
      3.4.1.1 Examine WSN Environments ............................... 54
      3.4.1.2 Examine BSN Environments ............................... 55
      3.4.1.3 Examine Security Protocols ............................... 55
    3.4.2 Modify/Propose Protocols .................................... 56
      3.4.2.1 Analyse Existing Protocols ............................... 56
      3.4.2.2 Modify Protocols .......................................... 58
      3.4.2.3 Propose Protocols ........................................ 58
    3.4.3 Implement/Evaluate ............................................... 59
      3.4.3.1 Security Analysis .......................................... 60
      3.4.3.2 Implement .................................................. 60
      3.4.3.3 Performance Memory Analysis ............................ 61
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.1</td>
<td>Security Goals in BSNs</td>
<td>113</td>
</tr>
<tr>
<td>5.4</td>
<td>Securing Sensors Measuring the Same SEVs</td>
<td>114</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Modified Encrypted Key Exchange Protocol using SEVs</td>
<td>115</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Proposed Protocol using SEVs for Nonces</td>
<td>117</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Attack and Security Weakness</td>
<td>118</td>
</tr>
<tr>
<td>5.5</td>
<td>Analysis of RSA and Elliptic Curves in BSNs</td>
<td>122</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Analysis of Other RSA based Password Protocols</td>
<td>122</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Comparison of RSA and Elliptic Curves</td>
<td>123</td>
</tr>
<tr>
<td>5.6</td>
<td>Anderson–Lomas Protocol</td>
<td>126</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Modified PPK Protocol</td>
<td>128</td>
</tr>
<tr>
<td>5.7</td>
<td>Securing Sensors Measuring Different SEVs</td>
<td>128</td>
</tr>
<tr>
<td>5.7.1</td>
<td>Proposed Three Party Protocol</td>
<td>132</td>
</tr>
<tr>
<td>5.7.2</td>
<td>Three Party Protocol with One Password Protocol</td>
<td>134</td>
</tr>
<tr>
<td>5.7.2.1</td>
<td>Attacks and Limitations in the Boyd Protocol</td>
<td>135</td>
</tr>
<tr>
<td>5.7.3</td>
<td>Proposed Protocol using the Boyd Protocol</td>
<td>135</td>
</tr>
<tr>
<td>5.7.4</td>
<td>Proposed Three Party Protocol with No Password Protocol</td>
<td>137</td>
</tr>
<tr>
<td>5.8</td>
<td>A Group Protocol</td>
<td>138</td>
</tr>
<tr>
<td>5.8.1</td>
<td>Proposed Group Protocol using SEVs</td>
<td>140</td>
</tr>
<tr>
<td>5.8.2</td>
<td>Comparison with Public Key Infrastructure</td>
<td>141</td>
</tr>
<tr>
<td>5.9</td>
<td>Analysis and Implementation</td>
<td>142</td>
</tr>
<tr>
<td>5.9.1</td>
<td>Implementation</td>
<td>145</td>
</tr>
<tr>
<td>5.9.2</td>
<td>Memory Size</td>
<td>148</td>
</tr>
<tr>
<td>5.9.3</td>
<td>Protocol Times</td>
<td>149</td>
</tr>
<tr>
<td>5.10</td>
<td>Conclusions</td>
<td>150</td>
</tr>
<tr>
<td>6</td>
<td>Validation</td>
<td>153</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>153</td>
</tr>
<tr>
<td>6.2</td>
<td>Use of GDM in Complex Systems</td>
<td>155</td>
</tr>
<tr>
<td>6.3</td>
<td>Analysis of the Home Health Care System</td>
<td>157</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Body Sensors</td>
<td>158</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Mobile Phone</td>
<td>159</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Home Sensors</td>
<td>159</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Health Controller</td>
<td>160</td>
</tr>
<tr>
<td>6.3.5</td>
<td>Other Components</td>
<td>160</td>
</tr>
<tr>
<td>6.4</td>
<td>Proposed Mechanisms for Key Establishment</td>
<td>160</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Initial Setup</td>
<td>161</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Home Health Controller and Body Sensor</td>
<td>161</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Body Sensors and Home Sensors</td>
<td>164</td>
</tr>
<tr>
<td>6.4.4</td>
<td>Mobile Phone</td>
<td>165</td>
</tr>
<tr>
<td>6.5</td>
<td>Modelling of the Home Health Care System using GDM</td>
<td>166</td>
</tr>
</tbody>
</table>
6.6 Specification of SAL ................................................................. 169
6.7 Validating Inter-Pulse Intervals .................................................... 170
  6.7.1 A Key Establishment Protocol using Physiological Values ........... 171
  6.7.2 Analysis of Entropy of Inter-Pulse Intervals ............................... 173
    6.7.2.1 Analysis of Different Inter-Pulse Interval Distributions ........... 174
    6.7.2.2 Amount of Entropy in a Single Day ................................. 176
    6.7.2.3 Worst Case Scenario When Using Entire Population .............. 178
    6.7.2.4 Worst Case Scenario using Individual Distributions ............. 180
  6.7.3 Problems with using SEVs with Error Correction ......................... 182
  6.7.4 Analysis of Inter-Pulse Intervals against Error Correcting Protocols 185
    6.7.4.1 Worst Case Scenario using Individual Distributions ............. 186
    6.7.4.2 Worst Case Scenario Excluding Same Inter-Pulse Intervals ...... 188
    6.7.4.3 Worst Case Scenario Excluding Half of the Same Inter-Pulse Intervals .......................................................... 190
  6.7.5 Protocol with Communication Overhead Removed ......................... 192
  6.7.6 Plaintext Attacks ............................................................... 195
  6.7.7 Implementation and Analysis .................................................. 196
    6.7.7.1 Android Smart Phone Results ........................................... 198
    6.7.7.2 Arduino Results ........................................................... 199
  6.8 Conclusions ............................................................................ 201

7 Conclusions .............................................................................. 205
  7.1 Contributions .......................................................................... 205
  7.2 Limitations ............................................................................. 207
  7.3 Future Research ...................................................................... 208

A Ethics Questions ........................................................................ 211
B Information Sheet ....................................................................... 213
C Consent Form ............................................................................. 217

Summary of Notation .................................................................... 219
Summary of Acronyms .................................................................. 221
List of References ........................................................................ 223
LIST OF FIGURES

1.1 Distributed Control System Flow Diagram ........................................... 2
2.1 A Sensor-Enabled Smart Home .............................................................. 11
2.2 Architecture and Service Platform of a Body Sensor Network (BSN) for Telemedicine and Home Health Care ..................................................... 13
2.3 Mapping of the Countermeasures .......................................................... 23
2.4 Electrocardiography Salient Features ................................................. 37
2.5 Light Absorption Through the Finger ................................................... 38
2.6 Different BSN Topologies ................................................................. 41
2.7 Single Frame using NV21 Format ....................................................... 43
3.1 Overall Methodology ........................................................................... 53
3.2 Investigation Phase of the Methodology ............................................ 54
3.3 Modify Propose Protocols Phase of the Methodology ...................... 56
3.4 Evaluation Phase of the Methodology ................................................ 60
3.5 Validation Phase of the Methodology .................................................. 62
4.1 Home Automation System ................................................................... 68
4.2 Filtering Single Server Security Protocols ......................................... 74
4.3 Number of Operations per Iteration ..................................................... 83
4.4 Number of Messages per Iteration ...................................................... 84
4.5 Number of Bits per Iteration ............................................................... 85
4.6 Number of Cycles Required for Different MAC Algorithms .............. 87
4.7 Number of Operations for Different Protocols ....................................... 96
4.8 Total Number of Bits Sent by Different Protocols .................................. 97
4.9 Number of Packets Sent by Different Protocols ................................... 100
4.10 Number of Bytes Sent by Different Protocols ..................................... 101
4.11 Total Time Taken by Different Protocols ......................................... 103
4.12 Total Time Taken on Single Network ............................................... 104
4.13 Total Time Taken using Different Networks ....................................... 104
5.1 Architecture and Service Platform of a BSN for Telemedicine and m-Health
5.2 Percentage of Invalid SEVs
5.3 Reading from the Sensor using the Serial Port

6.1 Home Health Care System Data Flow Diagram
6.2 Behaviour Tree Representation of the Health Care System
6.3 Intruder Represented as a Behaviour Tree
6.4 Mobile Phone Represented as a Behaviour Tree
6.5 Updated Behaviour Tree of a Home Health Care System
6.6 An Mobile Health System
6.7 The Different Distributions for Different People
6.8 The Different Distributions of the Entropy with IPI Values Concatenated
6.9 Using the Most Likely Result for the Low Entropy Key
6.10 The Different Distributions after Concatenating IPI Values
6.11 The Different Distributions after Concatenating IPI Values with the Error Margin
6.12 The Distributions after Concatenating IPI Values that are Different with the Error Margin
6.13 The Distributions after Concatenating Less Than Half the Same IPI Values with the Error Margin
6.14 The Probability of Not Getting Multiple Collisions using the ECCH
6.15 Using the Smart Phone to Obtain the Heartbeat
6.16 Light Intensity Measurements using the Smart Phone Camera on a Finger
6.17 Light Intensity Measurements using the Smart Phone Camera on a Finger
6.18 Light Intensity Measurements using the Arduino Device on a Finger
6.19 Arduino Device used to Measure Inter-Pulse Intervals
6.20 Voltage Output from Light Intensity Sensor on Finger
# List of Tables

2.1 Example Implementations of Home Health Care Systems .................................. 14  
2.2 Sensor Challenges ............................................................................................. 16  
2.3 Threats and Countermeasures on a Home Health Care System ......................... 24  
4.1 Notations ........................................................................................................... 70  
4.2 Comparison of Filtered Traditional Network Single Server Protocols ............... 75  
4.3 Performance Comparison of Single Server Protocols ......................................... 81  
4.4 Performance Comparison of Single Server Protocols ......................................... 82  
4.5 Memory Overhead in Bytes on MICA2 Platform .................................................. 85  
4.6 Performance Measurements for MSP1 .................................................................. 98  
4.7 Performance Measurements for MSP2 .................................................................. 98  
4.8 Performance Measurements for MSP3 .................................................................. 99  
4.9 Comparison of Multiple Server Protocols ............................................................ 101  
4.10 Ratio Compared to Exclusive-Or ......................................................................... 105  
4.11 Memory Overhead in Bytes on MICA2 Platform .................................................. 106  
4.12 Time Measurements for Different Algorithms ................................................... 106  
5.1 Body Sensor Data Rate Range ............................................................................. 111  
5.2 Notations ........................................................................................................... 112  
5.3 Comparison of Different SEV Strengths at 10 b/s ............................................... 121  
5.4 Comparison of Different SEV Strengths at 19.2 kb/s ......................................... 121  
5.5 Memory Overhead in Bytes on MICA2 Platform .................................................. 125  
5.6 Comparison of Password Key Establishment Protocols ....................................... 127  
5.7 Comparison of Existing Traditional Network Single Server Protocols ............. 130  
5.8 Number of Bytes Sent by Each Protocol ............................................................. 131  
5.9 Performance Comparison Lin vs. Proposed ......................................................... 143  
5.10 Simulation Results ............................................................................................. 145  
5.11 Memory Overhead in Bytes on MICA2 Platform .................................................. 146  
5.12 Comparison of Application Speed: As Ratio to Exclusive-Or ............................. 147  
5.13 Memory Size for Different Algorithms ................................................................ 148  
5.14 Time Measurements for Different Algorithms ................................................... 149
## List of Protocols

<table>
<thead>
<tr>
<th>Section</th>
<th>Protocol Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Gong’s simplified multi-server protocol</td>
<td>71</td>
</tr>
<tr>
<td>4.2</td>
<td>Chen-Gollmann-Mitchell multi-server protocol</td>
<td>72</td>
</tr>
<tr>
<td>4.3</td>
<td>Boyd key agreement protocol</td>
<td>75</td>
</tr>
<tr>
<td>4.4</td>
<td>Modified Boyd Key Agreement protocol</td>
<td>76</td>
</tr>
<tr>
<td>4.5</td>
<td>Bellare–Rogaway MAP1 protocol</td>
<td>77</td>
</tr>
<tr>
<td>4.6</td>
<td>NEKA Protocol — No Encryption Key Agreement Protocol</td>
<td>78</td>
</tr>
<tr>
<td>4.7</td>
<td>Attack on Boyd key agreement protocol</td>
<td>79</td>
</tr>
<tr>
<td>4.8</td>
<td>FPKDC — Five-pass KDC protocol</td>
<td>79</td>
</tr>
<tr>
<td>4.9</td>
<td>SPINS key agreement protocol</td>
<td>80</td>
</tr>
<tr>
<td>4.10</td>
<td>PIKE key agreement protocol</td>
<td>80</td>
</tr>
<tr>
<td>4.11</td>
<td>Attack on the PIKE key agreement protocol</td>
<td>81</td>
</tr>
<tr>
<td>4.12</td>
<td>MSP1 — Multiple Server Protocol 1</td>
<td>88</td>
</tr>
<tr>
<td>4.13</td>
<td>MSP2 — Multiple Server Protocol 2</td>
<td>91</td>
</tr>
<tr>
<td>4.14</td>
<td>MSP3 — Multiple Server Protocol 3</td>
<td>93</td>
</tr>
<tr>
<td>4.15</td>
<td>Multiple Server Protocol for SPINS</td>
<td>94</td>
</tr>
<tr>
<td>4.16</td>
<td>Modified Boyd protocol integrity change</td>
<td>95</td>
</tr>
<tr>
<td>5.1</td>
<td>Venkatasubramanian BSN protocol</td>
<td>113</td>
</tr>
<tr>
<td>5.2</td>
<td>Diffie–Hellman-based EKE protocol</td>
<td>115</td>
</tr>
<tr>
<td>5.3</td>
<td>Modified Diffie–Hellman-based EKE protocol</td>
<td>116</td>
</tr>
<tr>
<td>5.4</td>
<td>SEVN — protocol using SEVs for Nonces</td>
<td>117</td>
</tr>
<tr>
<td>5.5</td>
<td>SPEKE protocol</td>
<td>123</td>
</tr>
<tr>
<td>5.6</td>
<td>Modified EKE protocol with elliptic curve</td>
<td>124</td>
</tr>
<tr>
<td>5.7</td>
<td>Anderson–Lomas protocol</td>
<td>126</td>
</tr>
<tr>
<td>5.8</td>
<td>Modified Anderson–Lomas protocol to use SEVs</td>
<td>127</td>
</tr>
<tr>
<td>5.9</td>
<td>Modified PPK protocol to use elliptic curve</td>
<td>128</td>
</tr>
<tr>
<td>5.10</td>
<td>Abdalla three party protocol</td>
<td>129</td>
</tr>
<tr>
<td>5.11</td>
<td>Bellare–Rogaway 3PKDP</td>
<td>132</td>
</tr>
<tr>
<td>5.12</td>
<td>CSPK — Combined Symmetric Password Key protocol</td>
<td>133</td>
</tr>
<tr>
<td>5.13</td>
<td>ENEKA — Enhanced No Encryption Key Agreement protocol</td>
<td>134</td>
</tr>
<tr>
<td>5.14</td>
<td>PNEKA — Password No Encryption Key Agreement protocol</td>
<td>136</td>
</tr>
<tr>
<td>Section</td>
<td>Protocol Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.15</td>
<td>NEKAEV — No Encryption Key Agreement with Environment Values Protocol</td>
<td>137</td>
</tr>
<tr>
<td>5.16</td>
<td>BTP — Boyd Two-Pass protocol</td>
<td>138</td>
</tr>
<tr>
<td>5.17</td>
<td>Burmester–Desmedt generalised Diffie–Hellman with broadcasts</td>
<td>139</td>
</tr>
<tr>
<td>5.18</td>
<td>SEVGK — SEV Group Key protocol</td>
<td>141</td>
</tr>
<tr>
<td>5.19</td>
<td>Lin Protocol</td>
<td>142</td>
</tr>
<tr>
<td>6.1</td>
<td>Venkatasubramanian BSN protocol for mobile health care</td>
<td>162</td>
</tr>
<tr>
<td>6.2</td>
<td>Modified Diffie–Hellman-based EKE protocol for mobile health care</td>
<td>163</td>
</tr>
<tr>
<td>6.3</td>
<td>NEKA protocol for mobile health care</td>
<td>164</td>
</tr>
<tr>
<td>6.4</td>
<td>SEV-based EKE protocol</td>
<td>173</td>
</tr>
<tr>
<td>6.5</td>
<td>ECKE — EKE based protocol with error correction</td>
<td>182</td>
</tr>
<tr>
<td>6.6</td>
<td>Anderson–Lomas protocol</td>
<td>192</td>
</tr>
<tr>
<td>6.7</td>
<td>ECCH — Error Correcting Collisionful Hash Protocol</td>
<td>193</td>
</tr>
</tbody>
</table>
I am sincerely grateful to my supervisor Dr. Vallipuram Muthukumarasamy for his guidance and support throughout my part-time doctoral studies. His enthusiasm, fruitful scientific discussions and critical appraisal of my ideas, has been vital. I do appreciate his willingness and understanding of the difficulties with guiding a part-time PhD candidate. I wish to express my gratitude to my associate supervisor Dr. Steve Drew for his critical review of my thesis. His review had me thinking about the philosophy of research for the very first time.

I am grateful to Irene Blee for donating and presenting to me *The Jeffrey Blee Memorial Prize*, awarded to the most outstanding post-graduate student in the Institute for Integrated and Intelligent Systems. Her encouragement, understanding and support for post-graduate students is an inspiration.

I would like to thank Dr. Elankayer Sithirasenan for introducing me to the Requirement Behaviour Trees, and critical appraisal of my use for the Requirement Behaviour Trees. Also, I would like to thank the late Dr. Geoff Dromey for supplying the tools to create and check the Requirement Behaviour Trees.

I am deeply indebted to Patrick Crilly for sharing his programming experience with Arduino devices and Android phones, for his friendly conversations and many useful scientific discussions. I also like to thank Matt Moores for introducing me to the R Programming Language, and the many coffee breaks, where we discussed our research war stories.

I want to thank the anonymous conference and journal reviewers for their constructive reviews. Their comments were a great help to improve upon my research.

I would like to express my gratitude to Zahra Jadidi, Muhammad Usman, Sarah Liu, Kamanashis Biswas, Gavin Bray, Patrick Crilly and Liz Hughes for their thoughtful proofreading of some of the sections from this thesis.

Finally, I would like to thank the Free Software Foundation for developing tools that made my research easier. Some of these tools were not mentioned as part of my research, they include: *\LaTeX*; GNU Linux; R Programming Language; Gnuplot; Inkscape; LatexDraw. Also, I would like to thank the openclipart community for making their drawings freely available for use and modification. Many of my drawings are based on this community of artists.
List of Publications

The bibliographic details for the papers and any work published in the course of the research that is included in major part in the thesis itself are supplied below:

Journals


Book Chapters


Conferences


**Patents**


1

INTRODUCTION

1.1 Overview

As sensors become a part of everyday life, they can impact the way people live and interact. The different types of sensor environments range from sensors covering large areas, to many sensors in a small area [97]. The characteristics of the environments vary widely; the environments can be:

Transportation  Sensors can help with the management of traffic and the transportation of goods. This can include traffic lights, traffic controllers to manage traffic flow, or shock sensors, impact indicators for fragile goods.

Smart Buildings  Sensors can be used in a distributed control system to efficiently manage building automation. For instance, smart buildings allow people to visualise and manage air-conditioning, lighting, or other appliances in the building.

Environmental Monitoring  Environmental monitoring is the monitoring of the characteristics and quality of the environment. This can include continuous monitoring of a river system for pH level, dissolved oxygen, conductivity, turbidity and colour.

Health and Well-Being  The monitoring of health and well-being enables the delivery of health to a patient. This can include wearable wireless health sensors for remote bio-monitoring. Monitoring health and well-being of patients will enable the patients to live independently.

Water  The management of distribution of water to the populace has become important
as water has become a scarce resource. Water management includes planning, developing and distributing of water resources.

Each environment can be considered to be a distributed control system. Figure 1.1 is a common framework used by each environment. The Real World signifies the environment that needs to be managed. For instance, Real World could be a building, a patient, water ways, toll way, or a rainforest. To manage each system, there needs to be a Data and Measurement Platform. The platform is used to collect in situ continuous sensor data. The data is then sent to the Analysis Platform. The analysis of the data involves the assimilation, interpolation and explanation of the measurements. By the end of the analysis cycle, the system should have extracted high-quality trusted data. The high-quality trusted data is then sent to the Decision Model, where simulations are run and predictions are made. The outcomes or final decisions may be sent to the actuators in the system. Actuators can include air-conditioning, lights or even a person. The requirement for high-quality trusted data, places the need for security, information assurance and reliability into the system.

Another consideration is that future networks are increasingly becoming heterogeneous and highly dynamic. Cooperative networking paradigm envisages advanced wireless networks to cost effectively provide multimedia services and applications (incorporating
various sensors) anywhere, anytime. Self-organising, opportunistic networking tries to exploit resources (such as, bandwidth, memory and computing power) available in various devices and nodes in the system. This approach of moving away from exclusively relying on infrastructure networks raises serious challenges to overall reliability and security of the system.

As an example application, this thesis will concentrate on mobile health care systems [99]. A mobile health care system has the potential to significantly influence the way people live and interact. The financial burden on health care services due to the ageing population and the increase in chronic diseases has made mobile health care systems a low-cost and an attractive alternative to the traditional services. The evolution of Internet-of-Things as well as the pervasiveness of smart phones is increasing the potential for mobile health care systems to remotely manage the health of a patient or the elderly. Smart phones and small devices, such as body sensors, can be used to remotely monitor patients suffering from chronic diseases and allow them to have relatively independent lives. As the number of heterogeneous sensors increases, so will the amount of interactions between the sensors. A mobile health care system may require a degree of real-time monitoring or data collection. For instance, a medical emergency will require data sent to medical staff as quickly as possible, rather than the data sent after a few hours or days. The problem will be more complex if there is a requirement that commands sent to body sensors also need to be in real-time. If the system recognises a possible medical emergency, it may need to notify other devices immediately to start recording data or actuate (for example, insulin pump and defibrillator). A mobile health care system has some unique challenges not found in other sensor networks.

Security or information assurance is a very important criterion of a mobile health care system, and security of the devices and smart phones is important. For instance, a device that sends a command to several body sensors should have the transmission secured. The smart phone is a specialised device that the patient can carry with them and may be able to connect with the body sensors and the Internet. In the case where this system is part of an e-health system, the mobile phone may communicate with a central system. Depending on the health risks and privacy concerns of the patient, all of the information may not be transmitted to the central e-health servers.

There are two major networks that exist in a mobile health care system:

- **Wireless Sensor Network (WSN).**
- **Body Sensor Network (BSN).**

A WSN is defined as a heterogeneous system combining small, smart and cheap, sensing devices (sensors) with general-purpose computing elements. The bandwidth requirements of the sensors may differ considerably, with some sensors only needing to send a byte of data indicating an event, while other sensors may need to send a larger amount
of data, such as high-definition pictures of the surroundings. The characteristics of a WSN environment may be:

- A large scale environment, in some cases thousands of kilometres in size.
- Open environment, easily allowing an adversary to physically compromise sensors.
- Each sensor performs a single function.
- Made up of a multi-tiered network.

A WSN consists of a potentially large number of sensors; there may also be a few control nodes, which may have more resources. The functions of the control nodes can be:

- Connecting the sensor network to an external network.
- Aggregating results before passing them on.
- Controlling the sensor nodes.
- Providing services not available to a resource constrained environment.

A BSN environment is limited by the size of the body or limb. Thus the size of the sensors has to be small, and be able to perform more than one task, since the number of sensors a body can contain is also limited. The characteristics of a typical BSN environment are:

- The size of the network can be the size of the body or limb.
- The sensors are physically secured, especially if implanted in the body.
- The sensors may perform multiple functions.
- The sensors communicate over one network.

A mobile health care system is a complex area, with different types of networks. The next section will describe the research scope of this complex area.

### 1.2 Research Scope

For any practical deployments of a mobile health care system, an important consideration of the system architecture is security. Sensors require security for data fusion, location awareness, routing, authentication, privacy and integrity just to name a few. The building blocks for sensor security are cryptographic primitives and key establishment protocols. However, sensors in a mobile health care system introduce a difficult set of problems mainly due to small memories, weak processors, limited energy and small packet size. Thus only a very few conventional protocols can readily be used in sensor networks. Another major issue is that mobile health care systems may contain networks of different types, and each network type has different security characteristics and requirements. This thesis closely examines the currently available key distributions protocols, their strengths and limitations.
1.2. RESEARCH SCOPE

When WSNs are deployed, they have a likelihood of becoming the target of adversar-ies [98, 117, 181]. In an open environment, an individual device or an intermediate sensor node may not be completely and permanently trustworthy. If the device is physically secure, then that does not necessarily stop attacks since an individual device may be compromised remotely. To make a key distribution protocol work in an environment where sensor nodes do not trust an individual base station, an authentication scheme, which can be used with limited resources and can reduce the requirement for trusting servers, needs to be found.

The use of small keys is one of the security vulnerabilities addressed in this thesis. For performance reasons, sensors use small keys. However, small keys give an attacker a greater opportunity to compromise a sensor node. Thus, small keys will require frequent refreshing. Old keys can be used to generate new keys. But, if the old key is compromised, then the new key can easily be compromised. An efficient and scalable mechanism to freshen the keys between the sensors nodes is required.

Sensors found in BSNs may be operational for years, especially if the sensors are implanted in the body of an individual [71]. Many of the existing WSN key establishment schemes do not apply to the BSN environment. However, there are some similar characteristics between WSN and BSN environments. For instance, a mechanism to update security keys is an important area of research. Also, adding and removing devices securely from the sensor network is important.

Several researchers have used environmental data as the only source of secret information to establish keys between different nodes [14–16, 130, 173]. A major benefit of using environmental data for authentication is that body sensors can then detect if another sensor is on the same body. However, the approaches taken require the environmental data to be cryptographically random. The number of different types of environmental data that is cryptographically random is limited to only a few cases.

Proposed health care systems contain many different components and hence are inherently complex [99]. The complexity hinders security proofs and detailed analysis, so much so that theoretically proving that a protocol is secure within an entire system is rarely performed. A common method is to break the problem down to smaller problems and thus prove each sub-system is secure. An example of a sub-system is a security protocol between two body sensor nodes. Another problem is that complex systems rarely stay static; with new devices and algorithms a system may have different functionality from one day to the next. A proof that a protocol is secure for one system does not guarantee that it is secure on the other system. When data are used for multiple purposes and the environment is heterogeneous and complex, it becomes important from a security or information assurance point of view to have a formal methodology to validate the system. A formal methodology is also important to ensure that the information sent to medical staff and actuators to dispense medicine is accurate (secure), and the correct actions are
taken. The formal methodology has a requirement that it can model the security and privacy aspects as well as the assumptions and the application correctness.

The cryptographically secure key between the body sensor and the smart phone should be easily generated, without any intervention from the patient. Health sensors can use Inter-Pulse Interval or Heart Rate Variance as good sources for cryptographically random numbers and the physiological values can be used as a one-time pad. However, the amount of entropy that can be obtained from a small amount of physiological data is unknown. So even if protocols can be developed that use low entropy keys that change over time, the generation of the low entropy keys from physiological signals has yet to be attempted. It is apparent from extensive literature review (in the next chapter) that there is a lack of current knowledge around the generation of low entropy keys, hence it is unknown on what problems may occur when used in any new protocols that are created. Another limitation is that there is no literature on gathering low entropy keys by sensing physiological data using pervasive smart phones and low-cost devices, such as Arduino devices.

The problems and limitations found in both WSN and BSN environments give rise to several research questions based on key establishment that needs to be addressed. The problems are that:

- Key distribution nodes can become compromised.
- Initial security keys are hard to configure.
- It is difficult to validate security assumptions made by protocols in a complex system.
- Finding how low entropy data be collected from the body using low-cost and pervasive devices.

The solutions described in this thesis for the above problems are based on a number of major research areas within authenticated key establishment protocols and mobile health care systems. This thesis will:

- Create new authenticated key establishment protocols that handle one or more key distribution nodes becoming compromised.
- Create new key establishment protocols that use physiological data to set up initial security keys.
- Develop a mechanism using Genetic Design Methodology to validate security assumptions made by protocols in a complex system.
- Use a database of physiological data to discover a quick mechanism to obtain low entropy keys collected from the body.
- Use a smart phone and Arduino devices to measure physiological data from a body.

A secure mobile health care system will increase the adoption and the safety for the people using the system.
1.3 Thesis Outline

This thesis is divided into seven chapters as follows:

- Chapter 1 gives a brief introduction, and outlines the major objectives of this thesis.
- Chapter 2 describes the relevant background of the research topic. The threats and countermeasures to mobile health care systems are described. A mechanism to define a complex system, to measure physiological data and several sensor development platforms are also described. The specific security limitations for health monitoring are explained.
- Chapter 3 identifies the research questions in the context of the security limitations described in Chapter 2. Chapter 3 also defines the hypothesis, used to answer the research question, and then describes the methodology to validate the hypothesis.
- Chapter 4 reports the secure key establishment protocols to handle compromised key distribution nodes.
- Chapter 5 describes key establishment protocols that use changing low entropy data. Specific protocols are designed to establish keys for: two nodes reading the same low entropy key; two nodes reading different low entropy keys; many nodes reading the same changing low entropy data.
- Chapter 6 describes the validation process of the security assumptions made by protocols. It also validates that physiological data can be used as low entropy keys, using low-cost and pervasive devices.
- Chapter 7 contains the major contributions made by this research, and outlines the future directions in the research.
If I have seen further it is by standing on the shoulders of giants.
Isaac Newton (1642–1727)

2
BACKGROUND

This chapter provides a critical review of relevant research reported in the literature. Problems and limitations of the current systems will be described in detail. There are five major sections in this chapter. The first section covers security for a mobile health care system, where the security threats as well as the countermeasures are described. Section 2.2 describes commonly used tools to design a complex system and some of the difficulties when verifying a complex system. Section 2.3 examines available mechanisms to measure physiological data, and how to interpret the physiological data for security protocols. Section 2.4 investigates the main tools that are used in sensor networks when developing and evaluating sensor applications. Section 2.5 summarises the main problems and limitations that were identified in this chapter.

2.1 Security for Mobile Health care Systems

This section explains the issues related to the security of a mobile health care system, from information assurance and administration to authorisation, accountability and availability. This section gives a brief introduction to each of the security countermeasures. To gain an understanding of the level of difficulty in implementing security mechanisms, this thesis provides a detailed description of how to implement a key management protocol for sensors. The chapter begins with a brief description of the general architecture and service platforms used for health care sensor systems.

The ageing population and the increase in chronic diseases have placed a considerable
financial burden on health care services. Body sensors may play a vital role in reducing health care costs significantly. Sensors can be used to remotely monitor elderly patients suffering from chronic diseases and allow them to have relatively independent lives. However, the use of body sensor networks is inherently complex. For instance, an increase in blood pressure during exercise is normal. But blood pressure increases while at rest could mean a serious medical condition. Sensors may measure not only physiological values but also a range of other parameters such as body motions, which can lead to a number of different sensors needing to communicate with each other. As the number of heterogeneous sensors increases, so will the complexity of interactions between the sensors. Different sensors have different associated costs and privacy acceptance levels. For example, a sensor simply detecting light will have different costs to a sensor recording sound. However, less costly and more acceptable sensors can be used to detect a phenomenon before alerting the more costly sensors to start their monitoring.

In some cases large areas can be covered by relatively few sensors, in other cases large areas can be covered by many sensors. It depends on the sensor application [97]. Different environments have a range of varying characteristics. For instance, physical access to a sensor is easier for an adversary if the sensor is in an open area than for a similar sensor implanted in an individual's body. A number of protocols described in this chapter are designed for mobile health care systems. However, these protocols can also be applied to other environments that have similar security characteristics. Figure 2.1 gives a diagrammatic representation of a mobile automation system that can be used to monitor the elderly. The diagram shows the communication of the sensors with the central Home Health Controller (HHC). The home controller is a specialised device that is situated at home and is connected to the Internet. When this system is part of a health system, the HHC is a specialised device that sends the necessary information to the hospital. Depending on the patient's health risks and privacy concerns, all of the information may not be transmitted to a hospital. For instance, home sensors may be activated only if other sensors detect that there may be a medical emergency, such as the patient lying horizontal in the kitchen. Surveillance software can be used to detect if the patient is cleaning the kitchen or getting something from the ground, or if there is actually an emergency. If the system does detect an emergency, the hospital staff is notified; they examine the information and decide on the best course of action.

Other sensors that may work in conjunction with home sensors are body sensors. Some of the data recorded by body sensors include the heart rate, blood pressure, temperature, and blood oxygen level. In some scenarios when only looking at the amount of medical data to collect the data rate can be as low as 2 bps [11, 184]. However, other information sent with the message, such as the location of the sender node (8 bits), a message authentication code (in this case the size is defined to be the same as the size of the physiological data) and a counter to stop replay attacks (32 bits), raises the data rate to around 10 bps.
2.1. SECURITY FOR MOBILE HEALTH CARE SYSTEMS

Therefore, due to the number and size of messages, the data rate is measured in bits rather than bytes. In this thesis there is the assumption that the bit rate required is 10 bps to send secure data. Another type of sensor is a surveillance camera [77], and the data rate requirement for video streams is much higher than that of body sensors. A single camera normally requires a data rate of 1 to 4 Mbps. Providing secure data transfer between sensors is a requirement for the mobile health care system.

An additional advantage of enabling security in the mobile health care system is that secure smart data masking enables data sharing among institutions and reduces costs for duplicate tests. Privacy laws may prevent the complete medical records of an individual from being transferred between institutions. Health information collected from sensors needs to be secured, and in some countries (for example, the US) security is mandated [167]. Securing a mobile health care system becomes difficult, mainly because of the different requirements for various components. For instance, the sensors have considerably more resource constraints than do mobile phones, cameras or desktop computers. With differences in computing power, as well as differences in communication costs, different security protocols may be required for the entire system. For instance, an efficient key establishment mechanism specifically for body sensors was created using physiological data [173]. However, the mobile health system may send physiological data to medical staff or to an analytics engine [62]. The physiological data may also be sent to an actuator to release medicine into the body [62].

When the same physiological data are used for more than one purpose (and taking
into consideration the complexity of a heterogeneous environment), it becomes important from a security or information assurance point of view to have a formal methodology. A formal methodology is also important to assure that the information sent to medical staff and actuators to dispense medicine is accurate and that the appropriate actions are taken. The formal methodology has a requirement that it can model the security and the privacy aspects as well as the application correctness. A number of researchers have used environmental data as the only source of secret information to establish keys between body sensors [15, 130, 173]. The major benefit of using environmental data is that body sensors can use this information to authenticate that the other sensor is also on the same person and not another individual. However, these researchers have cited a number of problems with that approach. The problems include that:

- Only cryptographically strong environmental data can be used.
- The environmental values can become compromised, in which case the new session key is also compromised.

These problems limit the use of environmental data for establishing keys to only a few cases. The other difficulty the system encounters is in the key establishment scheme of the PDA with sensors at home and in the body. It is envisaged that the patients will simply be users of the system and will not be able to set up security certificates or keys.

This thesis shows protocols that address these problems. This thesis shows that password protocols can be used to establish keys between body sensors, if passwords are replaced by physiological data. A new protocol is developed to allow a patient to connect a PDA to the mobile health care system and thus be able to view information about each of the sensors (ranging from cameras to body sensors). The protocol does not require traditional encryption to transport the new session key. This thesis shows that the sensor nodes can establish keys even if no previous shared keys exist between them.

2.1.1 Mobile Health Care Sensor Systems

Health care sensor networks may contain many different types of sensors. The sensors in such a system can range from powerful high-definition cameras to low-powered Electrocardiography (ECG) body sensors. Figure 2.2 shows a patient at home with a number of body sensors that can communicate with a camera sensor, the HHC and a PDA. The cameras may start recording only if the body sensors detect that there may be a medical anomaly, such as a dramatic increase in blood pressure. Surveillance systems, such as S3 [73], can be used to detect if the patient is involved in an activity that would increase the blood pressure (such as exercise). If the patient is not involved in any activity that would account for the medical anomaly and the hospital staff are notified. They examine the information and decide on the best course of action. The PDA is used to give feedback to the patient about the condition of his or her body, as well as the status of the sensors.
The PDA can notify the patient of any detected emergency, allowing the patient to report back a false alarm if one has occurred. The PDA can be replaced with a mobile phone or any other hand-held communication device.

There are numerous types of communication technologies that are used in the mobile health care system. The communication technology can be:

- ANT+ (Active Network Technology).
- Bluetooth.
- Low Powered Bluetooth.
- ZigBee.
- Zwave.
- 802.11 (WiFi).
- 802.3 (Ethernet).

This list shows the need to inter-operate between the different technologies. Each technology has its own security risks, and combining technologies can increase the se-
CHAPTER 2. BACKGROUND

curity risks. A number of software and hardware architectures have been proposed and implemented to evaluate and address the needs of a mobile health care system. Table 2.1 summarises some of the major research projects into body sensor networks and monitoring [98, 117, 181].

Table 2.1: Example Implementations of Home Health Care Systems

<table>
<thead>
<tr>
<th>Platform</th>
<th>Base Device</th>
<th>Wireless Protocol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>HealthGear</td>
<td>Phone</td>
<td>Bluetooth</td>
<td>Wearable system for connecting sensors and mobile phones</td>
</tr>
<tr>
<td>CodeBlue</td>
<td>PC, PDA</td>
<td>802.15.4</td>
<td>Provision of medical monitoring in a hospital environment</td>
</tr>
<tr>
<td>ALARM-NET</td>
<td>PC, PDA, Stargate</td>
<td>Bluetooth, 802.11</td>
<td>Wireless sensor network for assisted-living and residential monitoring</td>
</tr>
<tr>
<td>DexterNet</td>
<td>PC, PDA, Phone</td>
<td>802.15.4</td>
<td>Body sensor network for indoor and outdoor monitoring</td>
</tr>
</tbody>
</table>

The wireless protocols used in existing mobile health care systems include a number of security provisions and options. Each protocol has many options and uses different security features. For instance, there are several vulnerabilities and pitfalls in 802.15.4 [136]. They categories can be:

- Initialisation Vector (IV) management.
- Key management.
- Integrity protection .

Bluetooth also has several vulnerabilities [66], which include:

- Bluejacking.
- Bluecasting.
- Bluesnarfing.

Several vulnerabilities have also been discovered in the 802.11 protocol over the last decade [27, 81]. They include:

- WEP cipher.
- Key management.
- A 40-bit secret key.

The wireless protocols themselves can have security vulnerabilities. Even with secure wireless protocols, the mobile health systems described in the preceding do not cover attacks such as impersonation of the user, modification of software or other higher-level attacks that are independent of the type of network that is used.
2.1.2 Compatibility Issues between Different Environments

The possible number of different networks and different types of environments are large. There are various types of sensor environments, ranging from sensors covering large areas [29, 115], to many sensors covering a small area [78, 137]. In a mobile health system major types of wireless sensor networks can be:

- Large area wireless sensor networks (*Home Sensor Networks*, HSNs).
- Small area wireless sensor networks (*Body Sensor Networks*, BSNs).

Table 2.2 compares HSNs and BSNs. The table is based on a comparison between BSN and a generic Wireless Sensor Network (WSN) [10]. Some of the more noticeable differences are the larger scale that exists in HSNs, compared to the severe constraints on the BSNs.

The different sensor environments have a wide range of different characteristics. For instance, sensors placed in a large open area are not as physically secure as sensors implanted in an individual’s body. This chapter discusses specific protocols for the BSN and HSN environments.

The protocols designed for BSNs can also be applied to other environments that have similar security characteristics to those found when using BSNs. The same is true for the protocols designed for the HSNs: they can be applied to environments which exhibit similar security characteristics.

2.1.3 Limitations with Power and Security

This thesis refers to a *sensor network* as a heterogeneous system combining small, smart and cheap sensing devices (sensors) with general-purpose computing elements. A sensor network consists of a potentially large number of sensors. Sensor network applications [35, 97, 173] include monitoring health, tracking bush-fires, monitoring wildlife, conducting military surveillance and monitoring public exposure to contaminants. A WSN may also contain smaller number of control nodes, which may have more resources. The functions of the control nodes include:

- Connecting the sensor network to an external network.
- Aggregating results before passing them on.
- Controlling the sensor nodes.
- Providing services (otherwise not available) to a resource-constrained environment.

Examples of control nodes [35] are the Stargate platform, the GNOME platform, the Medusa MK–2 platform, and the MANTIS platform. These platforms may use higher-level operating systems such as the Linux operating system. The platforms themselves may have additional communication mechanisms. For instance, the GNOME platform also has
## CHAPTER 2. BACKGROUND

### Table 2.2: Sensor Challenges

<table>
<thead>
<tr>
<th>Challenges</th>
<th>External Sensors</th>
<th>Body Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale</td>
<td>As large as the environment being monitored (metres/kilometres)</td>
<td>As large as the human body</td>
</tr>
<tr>
<td>Node Number</td>
<td>Greater number of nodes required for wide area coverage</td>
<td>Fewer, more accurate sensors nodes required</td>
</tr>
<tr>
<td>Node Function</td>
<td>Sensors perform a single dedicated tasks</td>
<td>A single sensor performs multiple tasks</td>
</tr>
<tr>
<td>Node Accuracy</td>
<td>Many nodes compensate for accuracy and allows result validation</td>
<td>Each sensor required to be robust and accurate</td>
</tr>
<tr>
<td>Node Size</td>
<td>Small size preferable but not needed in many cases</td>
<td>Need for miniaturisation</td>
</tr>
<tr>
<td>Conditions</td>
<td>Exposed to extremes in weather and environment</td>
<td>More predictable environment</td>
</tr>
<tr>
<td>Event Detection</td>
<td>Early adverse event detection desirable</td>
<td>Early adverse event detection vital</td>
</tr>
<tr>
<td>Variability</td>
<td>Ranges from static to mobile networks</td>
<td>Static network but limb movement can change the structure</td>
</tr>
<tr>
<td>Data Protection</td>
<td>Many cases only require authentication</td>
<td>Patient data needs to be encrypted and authenticated</td>
</tr>
<tr>
<td>Power Supply</td>
<td>Accessible and likely to be changed more easily and frequently</td>
<td>Inaccessible and difficult to replace in implanted sensors</td>
</tr>
<tr>
<td>Power Demand</td>
<td>Likely to be greater as power is more easily supplied</td>
<td>Likely to be lower as energy is more difficult to supply</td>
</tr>
<tr>
<td>Energy Scavenging</td>
<td>Solar, and wind power as candidates</td>
<td>Motion and thermal as candidates</td>
</tr>
<tr>
<td>Access</td>
<td>Sensors can be more easily replaced</td>
<td>Implanted sensors difficult to replace</td>
</tr>
<tr>
<td>Biocompatibility</td>
<td>Not considered in most applications</td>
<td>Needed for implanted sensors</td>
</tr>
<tr>
<td>Context Awareness</td>
<td>Not as important in static environment</td>
<td>Important because body physiology is sensitive to context change</td>
</tr>
<tr>
<td>Data Transfer</td>
<td>Loss of data compensated by number of sensors</td>
<td>Loss of data more significant</td>
</tr>
</tbody>
</table>
an Ethernet connection. It should be noted that a sensor node may also be a control node. Controller nodes are not limited to custom-made hardware. For instance, the UbiMon project [110] uses a PDA as a controller node, while other options for controller nodes include mobile phones [62].

Communication is the most expensive operation in sensor networks, where the received power drops off as the fourth power of distance mainly due to multipath propagation. If the distance between the nodes is 10 m, the energy required to send a single bit is equivalent to the energy required to perform 5000 operations. Therefore, when the distance between the nodes is 100 m, the energy required to send a single bit is equivalent to the energy required to perform 50 million operations. When information encrypted using symmetric cryptography is transmitted and received, only 2% of the energy consumed is used to encrypt or decrypt the data [127].

Mica sensor nodes run TinyOS [102], an event-driven operating system specifically designed for wireless sensor environments. The memory footprint for TinyOS is small; a minimum installation (the core components) uses 400 bytes of data and instruction memory. TinyOS is developed in nesC and supports other hardware platforms. The TinyOS network packet wraps the payload during sensor node communication. To reduce the message size, the packet header contains a minimal amount of information. For instance, it does not contain the source address of the sender.

To enable security in a mobile health care system, the sensors will require more power. Most of the power consumption will be due to the extra communication costs. For instance, without security mechanisms a message in the 802.15.4 protocol contains a Cyclic Redundancy Check (CRC) value of two bytes. When using an 802.15.4 protocol security suite, the CRC value is replaced by a Message Authentication Code (MAC) value. The smallest MAC size in the 802.15.4 protocol is four bytes. Every secured message will have an additional overhead. Also, additional security management protocols, such as key establishment protocols, need to added and run.

### 2.1.4 Information Assurance, Security and Privacy Threats

Security and privacy concerns are impediments to health systems. The lack of security and privacy could limit the adoption of a mobile health care system, either because of legal consequences or because of patient misgivings about allowing devices such as cameras into their homes. If a mobile health care system was adopted with a limited security, the patient would be put at risk. For instance, a device in the mobile health care system that dispenses medicine or measures important vital signs should be secured against attacks. Not only are security and privacy important, but information assurance is also just as important. Information assurance guarantees that the data are correct not only from an adversary but also from malfunctioning devices. Security and privacy threats in the system
include:

- Impersonation of the user.
- Impersonation of the service.
- Modification of software.
- Modification of data.
- Disclosure of sensitive data.
- Denial of service.
- Repudiation.

Threats may stem not only from adversaries but also from the users' expectations or the users themselves. The users of the system may be the elderly who may have difficulty learning new technology. The threats from users' expectations may be:

- Difficulty in using complex technology.
- Inability to keep track of changing technology.
- Lack of trust of the system.
- Expectation for reliability.
- Expectation for real-time communication.

A detailed description of each of the threats is described in the next sections. After the threats are defined and described the countermeasures for the threats will be described.

### 2.1.4.1 Impersonation of the User

An adversary may attack the system by taking on the identity of one of the users in the system. The users may range from patients to medical staff to computer administrators. Each type of user has specific privileges and permissions, which an adversary may utilise to gain access to the system.

If the adversary assumed the identity of a patient, incorrect information could be passed through to the medical staff. The adversary might also be able to access sensitive information that should be available only to the patient. An adversary who obtained the identity of a medical staff person or computer administrator could become even more dangerous. The adversary might then be able to obtain information about patients or affect their treatment.

### 2.1.4.2 Impersonation of the Service

A complex system such as a mobile health care system has a number of remote services. Services may include analyses of data, authentication server or sending an emergency call. An adversary successfully impersonating a service can have major consequences for
the integrity of the system. For example, if an adversary was able to impersonate an authentication server which leased session keys out to devices, the adversary would then have access to all the session keys within the system, and the devices in the mobile health care system could easily become compromised.

2.1.4.3 Modification of Software

Viruses, Trojan horses, root kits and software update failures are only some of the methods to modify the software in a mobile health care system. Once the integrity of the system is compromised, any data or information derived from the system is also compromised. As devices become more intelligent and complex, there will be more opportunities for an adversary to create more complex malware to manipulate the system.

2.1.4.4 Modification of Data

Another possible attack is to modify the data. The modification of data can either be at rest (stored in memory) or in motion (transmitted on the network). If the data obtained by the mobile health care system can be modified, then medical staff and the end user will have misgivings about how accurate the measurements are. The data may also be manipulated to produce false emergencies or even contribute to an unwanted emergency.

2.1.4.5 Disclosure of Sensitive Data

In reality privacy is a major concern for patients. In a traditional health care system, medical records have sensitive information that patients would not like the general population to know. Home health care systems add an extra dimension to the problem, as more sophisticated devices can record a patient’s every move. Some devices are not very intrusive, such as temperature sensors, but other devices, such as cameras, are far more intrusive. The ability for the patient control what data are disclosed is an important feature of a mobile health care system.

2.1.4.6 Denial of Service

Denial of service can be initiated by an adversary. Denial-of-service activities can range from wormhole attacks to draining the battery of a sensor by initiating unnecessary requests. If an adversary can easily create a denial-of-service attack and make the system unusable for an extended period of time, the end user may be placed in serious danger.
CHAPTER 2. BACKGROUND

2.1.4.7 Repudiation

Repudiation is the act where a participant in the system has forged that they have performed an action. For instance, a patient may forge that they had taken their medicine, and state that there must be something wrong with the health care system.

The hospital obtaining a guarantee a patient has taken his or her medicine and the patient obtaining a guarantee that a call sent has been received by the hospital are both important security requirements. In both cases the information assurance of the system increases, as well as its usability.

2.1.4.8 Difficulty in using Complex Technology

A complex system where a user needs to install security certificates is unfeasible. Setting passwords on tens to hundreds of devices is technically difficult due to the lack of user interfaces such as keyboards and monitors on many of the sensors. Forcing a patient to remember a new password or PIN will be a deterrent to the take-up of a mobile health care system. It is envisioned that the elderly could benefit from a mobile health care system. However, they would have the greatest difficulty in managing and handling complex technology.

2.1.4.9 Inability to Keep Track of Changing Technology

Changing the components in an ever-changing environment and then having the users learn that new technology is unfeasible. As technology improves, new devices will be added to an already-complex system. The initial security set-up should be simple so that any new devices can be easily added and old devices can be securely removed.

2.1.4.10 Lack of Trust on the System

The user needs to trust the system to use it. The user should be in control of the system, as well as the devices and disclosure of data. For instance, devices such as cameras capture sensitive data and therefore should turn on if there is an emergency or when there is a request from the user. The types of control and feedback that are given to the user will influence the adoption of a mobile health care system.

2.1.4.11 Expectation of Reliability

The quality of security services is a very important criterion of the mobile health care system. Security services should be available to facilitate the smooth operation of the system even if a component malfunctions or is compromised. As the number of components in a
mobile health care system increases, so does the likelihood of failure. The system should be designed so that the failure of a component does not jeopardise the security of the system. For instance, if the authentication server is unavailable, then the system should be able to cope, either with a redundant authentication server or with another device that would take on the new load until the authentication server was replaced.

2.1.4.12 Expectation of Real-Time Communication

The security mechanisms should enable real-time communication to and from the sensors. When an emergency occurs there is an expectation that information will be sent to the relevant authorities as soon as possible. In a medical emergency, time is a critical factor for the well-being of a patient.

2.1.5 Countermeasures to the Threats

This section describes security mechanisms to mitigate the threats and deterrents to the mobile health care system. When looking at countermeasures, Certified Information Systems Security Professional (CISSP) guidelines are examined. CISSP is an independent information security certification governed by the not-for-profit International Information Systems Security Certification Consortium. According to CISSP there are ten domains to security [86], they are:

- Information security and risk management.
- Access control.
- Telecommunications and network security.
- Cryptography.
- Security architecture and design.
- Operations security.
- Application security.
- Business continuity planning and disaster recovery planning.
- Legal, regulatory, compliance and investigation issues.
- Physical security.

The ten domains are based on three fundamental security characteristics of information: confidentiality, integrity and availability. Cryptographic keys are a cornerstone to confidentiality and integrity. The quality of security is a very important criterion of the mobile health care system.

Ensuring the quality of security services in a mobile health care system is difficult, mainly because of the number of different components, each with different characteristics. The mobile health care system's security services should be able to handle the failure of a
single or multiple components. Sensors, mobile phones, cameras and desktop computers have dramatically different resource constraints. The differences in computing power and communication costs require different security protocols throughout the entire system.

2.1.5.1 Privacy

Privacy is the ability for users to control their personal data. The users should be able to view and control their data. Privacy is enabled by securing data, data transmission and storage. There needs to be due diligence to ensure that no sensitive data are sent to the incorrect groups or individuals unless these have permission to see the information. Any failure will cause a lack of trust by the end users.

Also, it is important that sensitive data (such as from cameras) are recorded only when necessary. The system should allow the end user an opportunity to control the type of data that is sent. Input from the end user is an important consideration when looking at the overall privacy of the system.

2.1.5.2 Mapping the countermeasures

Table 2.3 maps the security and privacy threats with the appropriate countermeasures. An audit is an important countermeasure for most of the threats, and it enables accountability. The countermeasures range from low-level cryptography algorithms to high-level policy management. The difficulty with the countermeasures is that different components have different security requirements. For instance, key management on the HHC is different from the key management of an implanted health sensor found in the body. The HHC may need to manage a wide variety of keys, from small keys found in body sensors to certificates found in some of the cameras. Integrating the different security implementations is a major problem in a mobile health care system.

The security countermeasures can be grouped into the five As: authorisation, accountability, availability, administration and assurance. The Figure 2.3 maps the countermeasures associated with the five As. Also, the countermeasures are broken down further. For instance, the audit countermeasure is an important part of accountability, but two important subsets of audits are entity authentication and policy enforcement. Entity authentication is used to distinguish different services, nodes, components and users. When auditing an action it is important to know who performed that action.

Administration is another vital countermeasure. For instance, the installation and configuration of the system, if done incorrectly, can introduce zero-day security holes. This is an example of requiring a secure mechanism. However, if a secure mechanism is implemented in a user-unfriendly way, where the user needs to set up certificates, keys and passwords on each of the components, then the user will find the system hard to use.
2.1. SECURITY FOR MOBILE HEALTH CARE SYSTEMS

Figure 2.3: Mapping of the Countermeasures
TABLE 2.3: Threats and Countermeasures on a Home Health Care System

<table>
<thead>
<tr>
<th>Threats</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impersonation of the user</td>
<td>Authentication, Audit, User Administration, Trust Policy, Key Management, Intrusion Detection</td>
</tr>
<tr>
<td>Impersonation of the service</td>
<td>Authentication, Audit, Service ID Management, Trust Policy, Key Management, Intrusion Detection, System Integrity</td>
</tr>
<tr>
<td>Modification of code</td>
<td>Code signature, access control, Audit, System Installation, Configuration, Remediation, Antivirus, Intrusion Detection, System Integrity</td>
</tr>
<tr>
<td>Modification of data</td>
<td>Modification of the data and Data Protection (Integrity), Access Control, Audit, Access Control, Data Protection Policy, Intrusion Detection</td>
</tr>
<tr>
<td>Disclosure</td>
<td>Data Protection (Confidentiality), Access Control, Audit, Access Control, Data Protection Policy</td>
</tr>
<tr>
<td>Denial of service</td>
<td>Firewall, Service Continuity, Disaster Recovery, Routing Control, Topology Control, Rate/Resource Limits, System Integrity</td>
</tr>
<tr>
<td>Repudiation</td>
<td>Audit, Non-Repudiation, Audit policy, Non-Repudiation Policy, System integrity</td>
</tr>
</tbody>
</table>

With a more complex system the likelihood is higher that the user will produce an error. Without cryptographic keys and key management, most of the countermeasures described in Figure 2.3 will be ineffectual.

2.1.5.3 Cryptography

Several cryptography libraries using symmetric keys [95, 127] have been proposed. Much of the work on sensor protocols has used a symmetric key cryptography library. Recent work has shown that even asymmetric keys may be used in WSNs [116, 178]. However, they still consume considerably more resources than the symmetric counterparts.

2.1.5.4 Key Management

Key establishment protocols are used to set up shared secrets between sensor nodes, especially between neighbouring nodes. When using symmetric keys, the key establishment protocols in WSNs can be classified into three main categories: Pair-wise schemes; Random key predistribution schemes; Key Distribution Centre (KDC). Recently, a fourth category, where key establishment protocols use environmental values, has been developed [130, 173]. This section discusses limitations and problems with existing key establishment protocols when applied to wireless sensor networks.
Many of the existing key establishment protocols are designed for open environments where a sensor node can be easily compromised [108]. Also, sensors may be short-lived and thus updating keys may not be important. This thesis examines traditional sensor key establishment protocols in the BSN environment as well, and show their limitations and problems.

2.1.5.5 Pair-wise Key Establishment

The first category of key establishment protocol is called pair-wise scheme. The simplest of this category is the full pair-wise scheme [41, 60], where each node in a network of total \( n \) nodes shares a unique pair-wise key with every other node in the network. Each sensor node will have a memory overhead of \((n - 1)\) cryptographic keys. Other pair-wise schemes [25, 101] also have \( O(n) \) memory cost. In a pair-wise scheme, the sensor network will not be compromised even if a fraction of the sensors are compromised.

However, because of the memory constraints found in sensors, predefined keys may not be practical in large sensor networks. Also, pair-wise key establishment is difficult to maintain if the sensor network is dynamic in nature, with additions and deletions of sensors. Furthermore, there is no secure mechanism available to update the keys. Keys in sensor networks are usually 64 bits in size, and they may become easily compromised. A server-less or server-based key establishment protocol can be used to update the key [31, 87, 89]. If the key between the sensors nodes is compromised before the running of the update protocol, then the new key will be compromised. If the key between the sensor nodes is compromised after the running of the update protocol, then the new key will be compromised, provided the messages from the update protocol were saved by the adversary.

Sensors found in BSNs may be operational for years, especially if the sensors are implanted in the body of an individual. Pair-wise key establishment is, therefore, not suitable if the keys need to update.

2.1.5.6 Random Key Establishment

Random key predistribution schemes are the second category of the key establishment protocol [41, 57, 60, 109]. This is a major class of key establishment protocols for sensor networks. They rely on the fact that a random graph is connected with high probability if the average degree of its nodes is above a threshold. After the connected secure network is formed, the protected links can be further used for agreeing on new keys, called path-keys. The random key establishment schemes have a security concern when a certain number of sensor nodes become compromised. If so, the entire network can then be compromised.

Another issue is that random key predistribution schemes require \( O(n) \) of predefined
data, and it is a major research area to limit the amount of predefined data [42]. The random key establishment schemes are designed for a large sensor network, and assume that sensors can be compromised. These characteristics do not match that of the BSNs. The BSN topology may not be suitable for random key establishment schemes. In BSNs, some sensors may only be able to communicate with one other sensor node. Another drawback is that the shared keys cannot be used for entity authentication, since the same keys can be shared by more than a single pair of nodes [72]. Random key establishment protocols also have no mechanism to update the keys.

2.1.5.7 Key Distribution Centre Schemes

When using a KDC scheme, if two entities, sharing no previous secret, want to communicate securely with each other, they generally do so with the assistance of a third party. In WSNs the two entities are typically resource-constrained sensor nodes, and the third party is a resource-heavy base station [127]. The base station provides an authentication service that distributes a secure session key to the sensor nodes. The base station is sometimes referred to as a trusted third party, since every client has to trust the base station by sharing a secret with it. The level of security of a typical key distribution protocol depends on the assumption that the authentication server is trustworthy [18, 21, 30, 38].

KDC schemes use the least amount of memory compared with the other two categories, and have an extra advantage of providing authentication for the sensor nodes [72]. Examples of KDCs in WSNs were first proposed in the Security Protocols for Sensor Networks (SPINS) protocol [127]. A simple example of a sensor environment may consist of a large number of motion sensors, a camera, and a base station. When the motion detectors are triggered, they notify the camera to start recording. If the motion sensors need to communicate to the camera via the base station, this may place undue stress on the network, and cause undesirable latency. To enable secure communication, the motion detection sensors will need to establish a shared secret between themselves and the sensor on the camera. Motion detectors may also be added, removed or replaced, thus adding the requirement to have a scheme that allows for a dynamic reconfiguration of a sensor environment.

Another use of a KDC scheme is when a sensor network is created using a random key predistribution scheme. If there is an increase in area that the sensor network needs to cover, a new set of sensors is distributed. To enhance the security of the expanded network, new values are used in the random key predistribution scheme for the added sensors. The new sensors will have to use a KDC scheme if they want to create a key between themselves and the old sensor nodes. The old sensors may have deleted the random key predistribution scheme from memory, so they can increase the amount of memory available to the node (and to increase security). If new nodes are added, they
will not be able to use the old scheme to create a key. Thus the SPINS protocol may not be suited for every WSN topology. For instance, it does not easily scale to a large WSN, since the non-uniform communication will focus the load onto the KDC. This may cause the battery life of the network to diminish considerably. Updating keys using a KDC mechanism can also cause concerns if the keys between the sensor node and the KDC become compromised.

Hybrid schemes have been created to offset this problem by combining different key establishment categories. The Peer Intermediaries Key Establishment (PIKE) scheme [42] is one such implementation. It combines a pair-wise scheme with the KDC scheme, in which one or more sensor nodes act as a trusted intermediary to facilitate the key establishment. The scheme was developed to limit the amount of memory used by the pair-wise and random key predistribution schemes and also to limit the communication load of the KDC schemes.

However, some of the limitations with using a sensor node as the trusted third party are that:

- The KDC scheme relies upon other schemes to create the trusted intermediary.
- The key sizes in sensor nodes are not large enough, so over a period of time, the key between the sensor and the trusted intermediary may become compromised. If the KDC protocol messages were captured and saved by an adversary, then the adversary may calculate the new keys created.
- Some sensor networks may not need an encryption algorithm; however, KDC protocols require an encryption algorithm to encrypt the new key.

The protocols may not be suitable for all the scenarios involving key establishment in sensor networks.

2.1.5.8 Environment Information

This thesis introduces the term Secure Environmental Value (SEV) to refer to sensed data that can be obtained by sensors from their environment. This data is usually hard to obtain through other means. Examples of environment where SEVs may be found include:

- The human body, where it is difficult to attach a device on the body without the person’s knowledge.
- A secured location, for instance, a military base or unmanned vehicle, such as Unmanned Aerial Vehicles (UAVs).
- Hard to reach places, for instance, a satellite in orbit.

The example environment used in this chapter is the human body where BSNs have been developed to measure the physiological values found in individuals [10]. Recent research in BSNs has shown that environmental information found in the body can be
used to secure communication between sensors nodes [14–16, 130, 173]. Health sensors can use Inter-Pulse Interval (IPI) [130] or Heart Rate Variance (HRV) [16] as good sources for cryptographically random numbers and the physiological values can be used as a one-time pad. Protocols that used these physiological values to encrypt a new key between a sensor pair have been developed [14, 173]. For instance, Venkatasubramanian BSN protocol used a single message to send a new key to the neighbouring sensor node.

Venkatasubramanian and Gupta noted that finding additional cryptographically sound physiological values is still an open research problem [173]. Another problem is that all the protocols developed with physiological values require all the sensor nodes to be able to measure the same phenomena. Only cryptographically strong physiological values (IPI and HRV have up to 80 bit Hamming distance between two different individuals [16]) can be measured. Also, developing technology (ultra wideband, UWB, radar [162]) may be used to remotely capture the heart rate and could cause security risks when only using IPI and HRV to secure the communication. Other cryptographically weaker physiological values, such as blood pressure, and iron count, are less susceptible to those remote attacks. The use of PIN code or a password is not applicable to BSNs since many of the sensors do not have a user-interface. Sensors also may be placed in hard to reach places, with some of the sensors implanted into the body. Also, the sensors may harvest energy directly from the body [94], allowing the sensors to exist for long periods of time. Key establishment protocols that increase the number of available environmental or physiological values will enhance this research area.

2.1.5.9 Reliable Security

Several researchers have addressed security of the controller nodes and/or the base station. SIA [132] addresses the issue of compromised nodes by using statistical techniques and interactive proofs, ensuring the aggregated result reported by the base station is a good approximation to the true value, even if a small number of sensor nodes and the aggregation node may have been compromised. However, the communication overhead between sensor nodes and the base station is high. Other works have shown that some of the statistical methods used are not resilient to a group of malicious sensor nodes, and the end user should be aware of which statistical methods are easily cheated [174]. Another way to protect results is to use a witness node mechanism [56, 183, 187].

A different approach is to protect the base station location. Routing mechanisms to protect the location and disguise the identity of the base station have been proposed [51]. Hop-by-hop re-encryption of each packet's header and data fields is designed to change the presentation of a packet so that it cannot be used to trace the direction towards or away from the base station. Uniform rate control is advised so that traffic volume nearer the base station is undifferentiated from traffic farther from the base station. Time decorrelation
between packet arrivals and departures further increases the difficulty of tracing packets.

However, ensuring that the authentication services are not hindered by a compromised or broken controller node or base station presents different challenges. A simple approach is to replicate the authentication services of the server so that any one of several servers can perform authentication. However, this approach reduces the level of security; if one server is compromised, security for every replicated server is compromised.

To make a key distribution protocol work in an environment where sensor nodes do not trust an individual base station, an authentication scheme needs to be developed, which can be used with limited resources and can reduce the requirement for trusting servers.

### 2.1.5.10 Group Keys

One of the best known available group key protocol, which many other group key protocols are based on, is the Microtimed Efficient Stream Loss-Tolerant Authentication (µTESLA) protocol designed by Perrig et al. [127]. In µTESLA, when a message is broadcast, it generates a message authentication code for that message with a symmetric key. The key is a number on a One-way Hash Chain (OHC). Each sensor is preconfigured with the initial number of the OHC, called the commitment, so that when the key is released, each sensor node can verify whether the key is valid. Only one sensor (in most cases it will be the base station) will have the entire OHC. To securely broadcast a message, the base station sends the message and its message authentication code generated by the key. At a later time, the base station releases the key to all the sensors. The sensors will first verify that the key is on the OHC, and then it can confirm that the original message is correct.

There are a number of problems with this approach. The mechanism is inefficient if there are many broadcast messages lost, sensor nodes will then spend a long time verifying the key against the one-way hash function. Another problem is that when an important event occurs, and a command needs to be broadcast out to all the sensors to increase their sampling rates, the µTESLA delayed approach does not perform well. The command that was broadcast cannot be verified until at a later time. Another problem is that all the sensors will have to be preconfigured with the initial number of the OHC, which makes it difficult to add or replace sensors to the system.

Recent research in BSNs has shown that environmental information found in the body can be used to secure communication between sensors nodes [14–16, 130, 172]. Body sensors can use Inter-Pulse Interval (IPI) or Heart Rate Variance (HRV) as good sources for cryptographically random numbers and the physiological values can be used as a one-time pad. Protocols that used these physiological values to encrypt a new key between a sensor pair have been developed [173]. For instance, Venkatasubramanian et al. [173] used a single message to send a new key to the neighbouring sensor node.

Only cryptographically strong physiological values, such as IPI and HRV, can be used.
CHAPTER 2. BACKGROUND

However, the method used to generate a strong cryptographic key can be more than one minute, since 67 quantised IPI values are required. Another problem was that physiological signals measured from different areas of the body have similar trends but not the same values.

Venkatasubramanian et al. [172] developed PSKA, as a mechanism such that the duration of the physiological signal capture is kept minimal. They used a vault [93] for the key agreement mechanism, where the physiological data is used to lock and unlock the vault. However, the scheme requires approximately 50 KB of memory and 10 KB of transmitted data. Another limitation is that if a body sensor is lost or stolen, the contents of the vault are no longer secure. Some of the body sensors will not have enough resources, and may not be physically secured to be able to use the PSKA scheme.

Another mechanism to create a secure communication between body sensors is to place a small electrical charge around the body and use that as the communication medium [63]. However, the sensors themselves will become more complex as more components are added. Also, the network will become insecure if two body sensor networks were to come closer and intersect.

An approach for a secure group key generation for off the shelf sensors with minimal memory and bandwidth resources is required.

2.1.6 Smart Cards and RFIDs in Mobile Health Care

There are four main papers concerning RFIDs.

Ren et al. [134] provided an overview of a mobile health care system. Four major problem areas were listed:

- The quality and reliability of patient monitoring.
- The power management of patients' devices.
- Context awareness.
- Security and privacy.

Ren et al. states the typical concerns in the area of security are:

- To prevent the disclosure of a patient's data.
- Who should have the right to access the patient's medical record.
- How to protect the privacy of the patient.

Ren et al. suggests the use of ECC to reduce the computational costs of public key cryptography. Key management is a critical part of a mobile health care system, since there is a large variation of keys available in such a system. Also, to encrypt a patient's information, a smart card is used to protect a patient's privacy and verify the patient's biometric information.
2.1. SECURITY FOR MOBILE HEALTH CARE SYSTEMS

Other areas of research [13, 171] were using electrocardiogram (EKG) and ECG as sources of randomness and as entity authentication as a viable mechanism to protect a mobile health care system. Another important security area in mobile health care systems is a trust mechanism [28], where historical behaviour is used to evaluate a device’s current trust. Ren et al. also suggests that Quality of Privacy (QoP) is another important criteria when investigating a successful mobile health care deployment.

Ameen et al. [6] lists several threats that can apply to a mobile health system, they are:

- Data modification.
- Impersonation attack.
- Eavesdropping.
- Replaying.

To counter the threats the following security measures were used:

- System Security.
  - Administrative Level Security.
  - Physical Level Security.
  - Technical Level Security.

- Information Security.
  - Data Encryption.
  - Data Integrity.
  - Authentication.
  - Freshness Protection.

Ameen et al. also argued that privacy is important in a mobile health care system. Processes and guidelines need to be in place, and circumstances involving emergencies should be considered. Apart from the processes, some other measures should also be implemented, they include:

- All communication needs to be encrypted.
- Users should not be specified unless required.
- Education on the importance of security and privacy.

Doss et al. [54] proposed an approach for authentication and privacy in RFID systems based on the minimum disclosure property. The system has three distinct components, the RFID, the RFID reader and the database. Different schemes are required if the channel between the reader and database is insecure. It is always assumed that the channel between the reader and RFID are insecure.

To show the security correctness of the protocol, formal proofs were used. The formal proofs chosen was based on proofs used by other research in the same field [44, 46–48]. The security correctness of the proposed protocols prove if the protocols achieve:
• Tag anonymity.
• Tag location privacy.
• Forward secrecy.
• Privacy.
• Resistance to replay.
• Resistance desynchronisation attacks.

Doss et al. [53] also proposed a secure mechanism to transfer ownership based on quadratic residues of a RFID tag. A property of the approach is that ownership transfer is guaranteed to be atomic. The scheme is protected against desynchronisation, which leads to permanent DoS. The scheme is suited to the computational constraints of passive RFID tags as they do not employ expensive hash functions and encryption functions. The security correctness of the protocol is based on the Avoine model [9]. The security analysis can show that the protocol can provide strong privacy.

An aspect of RFID security is secure ownership transfer as a tagged product changes control over the distribution chain. As a tagged product moves through a system, such as from the manufacturer to the distributor, to the retailer and then to the customer, ownership changes multiple times. A requirement for such a system is that the internal state of the tag reflects these changes. Ownership transfer guarantees only the new owner is able to interrogate the tag and the previous owner is prevented from communicating with the tag.

A formal definition of ownership and ownership transfer is defined by van Deursen et al. [168]. Formal methods to verify the correctness of privacy in security protocols also exist [9, 79, 170].

2.2 Design and Verification of Complex Systems

This section describes commonly used methods and tools to design a complex system and some of the difficulties when verifying a complex system.

2.2.1 Formal Verification

A formal method to verify that a protocol is correct is an important area in the research community. Verifying a protocol provides the validity that the protocol correctly does what it was meant to do and hence it is a significant step in analysing the protocol. The complexity of security protocols makes their verification a difficult task. Informal qualitative arguments by themselves are not reliable or acceptable, thus a formal analysis to verify the claim made by a protocol is needed.
Computer assisted formal methods for verifying security protocols can be divided into two major categories, which are:

- **Model Checking**: considers a finite number of possible protocol behaviours and allows checking if that satisfies a set of correctness conditions. This method works well for finding attacks on a protocol, rather than proving their correctness [50, 112, 122].

- **Theorem Proving**: considers all possible protocol behaviours, and checks that they satisfy a set of correctness conditions. This method works well for proving protocol correctness, rather than finding attacks on protocols [119, 125, 161].

Both model checking and theorem proving methods require computer assistance to aid with the analysis. However, methods based on theorem proving are less automated than those based on model checking.

A useful feature of model checking methods is that they can prove an attack when a protocol does not satisfy a correctness condition. The failure to find an attack implies that the protocol is correct. However, model checkers do not provide a symbolic proof that can explain why a protocol is correct and thus are uninformative when checking a valid protocol. Another important limitation of model checking methods is that they only guarantee correctness of a scaled down version of the protocol.

Theorem proving mechanisms have their own strengths and limitations. One of the strengths of theorem proving methods is that they can provide a symbolic proof when a protocol is found to be valid. Their main limitation is that they generally require more expert human guidance than methods based on model checking.

Another mechanism to verify that a protocol is secure is to use a mathematical proof [39]. Problems with using mathematical proofs include:

- With each small change in the protocol a new proof needs to be constructed.
- Security proofs are complex and involve long mathematical reasoning and are difficult to understand to the average practitioner.
- There are relatively few protocols with mathematical security proofs.
- As systems become more complex, constructing mathematical proofs becomes more challenging.

A combination of informal verification, machine analysis (either using model checking, or theorem proving), and mathematical proofs is important to gain assurance on the security of the protocol. The following section describes how Genetic Design Methodology (GDM) is used when examining complex systems and the correctness of security protocols.
2.2.2 Use of GDM in Complex Systems

The common techniques used to verify that a protocol is correct do not easily scale to a complex system. Using the above techniques to validate a system with hundreds of nodes, and many different protocols, is almost unfeasible. To further complicate matters, there are inherent restrictions such as; the formal verification will need to be repeated for every minor change in the system.

When proving that a protocol is secure, the proof relies on a number of assumptions made about the environment where the protocol is run. This may be assumptions such as secure time synchronisations between the parties, or that the physical security of the communication medium. Showing that a protocol is secure in a complex system may be considered as a two step process. The first step is proving that the protocol is secure based upon some assumptions. The second step is to show that the assumptions are valid and consistent in a complex system. The GDM has recently been used as a tool to prove and/or validate the overall correctness of complex systems.

Sithirasenan et al. [160] have used GDM to check the correctness of the 802.11i wireless security protocol. The requirements of the protocol were placed into a number of Requirement Behaviour Trees. The requirements were then verified by integrating them into a single Integrated Behaviour Tree. Thereafter, the Behaviour Tree model was translated into SAL formal notations for theorem proving. This mechanism shows that both model checking and theorem proving approaches can be performed using the same analysis tool. The checks performed was mainly focused on the protocol correctness and not on the assumptions made by the protocol.

When using GDM, systems are designed out of the requirements as opposed to methods that produce designs to meet the requirements. A major advantage of GDM is that it produces graphical models that are derived and integrated from the original requirements. The models can easily be used to verify that security protocols correctly work in a complex system.

An example of a complex system is the mobile health care system. For instance, in a mobile health care system it can become difficult to track how sensed data is used at different stages in the system. When the sensed data is also used in key establishment protocols, tracking the various uses of sensed data becomes even more important. For example, some key establishment protocols require that the sensed data never to be sent in the clear or to an untrusted third party, whereas other protocols do not need such restrictions. The complex system and the protocols can be defined in requirement behaviour trees using GDM [160].

Each requirement can be represented as a behaviour tree; this representation is specifically called a Requirement Behaviour Tree (RBT). An important part of the genetic design methodology is constructing the behaviour trees. Dromey [55, 180] defined Behaviour
Trees as: a formal, tree-like graphical form that represents behaviour of individual or networks of entities which realise or change states, make decisions, respond-to/cause events, and interact by exchanging information and/or passing control.

Behaviour trees provide a direct and clearly traceable relationship between what is expressed in the natural language representation and its formal specification. Conventional software engineering method applies the underlying design strategy of constructing a design that will satisfy its set of functional requirements. Whereas, a clear advantage of the behaviour tree notation is that it allows the construction of a design out of its set of functional requirements, by integrating the behaviour trees for individual functional requirements (RBTs), one-at-a-time, into an evolving Design Behaviour Tree (DBT).

The RBTs are integrated based on the precondition of the tree that must be satisfied in order for the behaviour encapsulated in a functional requirement to be accessible or applicable or executable. The resultant tree is called the Integrated Behaviour Tree (IBT). If there is no matching post-condition embodied in the evolving DBT, then a defect is identified and needs to be rectified. In which case, either the requirement is invalid, or there is a missing requirement. Integrating RBTs is an important feature when showing that security requirements in the system are valid or if there is a missing security requirement.

Once the RBTs are integrated, and any missing or invalid requirements are dealt with, then other models can be generated from the evolved DBT. SAL code can be generated; allowing the creation of theorems that also checks the security requirements of the system.

Behaviour trees can in turn be used to generate SAL code [160]. A model checker can then be used to verify the SAL code and thus verify the protocol in the sensor environment. The main steps in the GDM are: translation of requirements to behaviour trees; integration of behaviour trees; architecture transformation; component behaviour projection; and component design. When modelling the entire system, genetic design has significant advantages over Unified Modelling Language (UML), state charts or other methods [55]. The advantages include:

- Allow designers to focus on the complexity and design of individual requirements while not having to simultaneously worry about the details in other requirements. The requirements can be dealt with one at a time (both for translation and integration).
- The component architecture and the component behaviour designs of the individual components are emergent properties of the design behaviour tree.
- The methodology concentrates on discovery of behaviour gaps, which in turn discovers requirement and security gaps. The focus of direct translation of requirements to design makes it easier to see and find gaps either manually or using automated tools.
Present an automated method of mapping changes in requirements to changes in design.

The use of GDM specifically in the security field has not been explored. Further research in this field is required to check the suitability of using GDM on security protocols. In particular, this thesis will show that the GDM analytical tool can effectively perform model checking on the correctness of protocol assumptions in a complex system.

2.3 Measuring Physiological Data

As described in Section 2.1.5.8, physiological data can be used as SEVs in security protocols. This section describes two types of physiological signals: ECG; and, photoplethysmographic. The methods to measure the physiological signals will also be briefly described. Both methods allow a device to detect heart beats.

2.3.1 Electrocardiography

An ECG is measured by a device that detects and amplifies the tiny electrical changes on the skin caused by the heart muscle depolarising during a heartbeat [80]. At rest, each heart muscle cell has a negative charge across its cell membrane. An influx of positive charges causes the depolarisation, which will activate the cell mechanisms to contract. A healthy heart will have an orderly progression of a wave of depolarisation. The depolarisation is detected, by an ECG device, as tiny rises and falls in the voltage between two electrodes placed either side of the heart. A typical ECG tracing of a heartbeat consists of a P wave, a QRS complex, a T wave, and a U wave, which is normally visible in 50 to 75% of ECGs. Figure 2.4 shows a typical ECG trace.

A brief description of the different parts of an ECG trace is:

**RR interval** The time interval between the peak of an R wave and the next peak of an R wave. The normal resting heart rate is between 60 and 100 beats per minute, which is around 0.6 to 1.2 seconds.

**P wave** During normal atrial depolarisation, the main electrical vector is spreads from the right atrium to the left atrium. This turns into the P wave on the ECG and for a resting heart rate it is around 80 ms.

**PR interval** The PR interval starts from the beginning of the P wave to the beginning of the QRS complex. The PR interval reflects the time the electrical impulse takes to travel from the sinus node through the AV node (the AV node is designed to delay the electrical impulse) and entering the ventricles. The PR interval is, therefore, a good estimate of the heart's function to delay the electrical impulse (otherwise the chambers will contract at the same time). Normally this delay is 120 to 200 ms.
2.3. MEASURING PHYSIOLOGICAL DATA

**Figure 2.4: Electrocardiography Salient Features**

**PR segment** The PR segment connects the P wave and the QRS complex. The impulse vector is from the AV node to the Purkinje fibers. This electrical activity does not produce a contraction directly and is merely travelling down towards the ventricles, and this shows up flat on the ECG. The segment is around 50 to 120 ms for a resting heart rate.

**QRS complex** The QRS complex is the result of the depolarisation of the right and left ventricles. They have a large muscle mass compared to the atria, so the QRS complex usually has much larger amplitude than the P-wave. For a resting heart the time is around 80 to 120 ms.

**J-point** The point where the ST segment begins and the QRS complex finishes.

**ST segment** The ST segment connects the QRS complex and the T wave. The ST segment is the result of the depolarisation of the ventricles. For a resting heart rate it will take 80 to 120 ms.

**T wave** The T wave is the result of the repolarisation of the ventricles. For a normal resting heart rate it is around 160 ms.
CHAPTER 2. BACKGROUND

**ST interval** The ST interval is measured from the J point to the end of the T wave. For a resting heart rate it is around 320 ms.

**QT interval** The QT interval is measured from the beginning of the QRS complex to the end of the T wave. It is up to 420 ms in heart rate of 60 bpm.

**U wave** The U wave is often completely absent. It is hypothesised to be caused by the repolarisation of the interventricular septum. They always follow the T wave and also follow the same direction in amplitude.

**J wave** The J wave is also often completely absent. It is an elevated J-point and appears as a late delta wave following the QRS or as a small secondary R wave.

The term *lead* can have several meanings in electrocardiography. The term lead may be used to indicate the electrical cable attaching the electrodes to the ECG recorder. Alternatively, the word lead may refer to the tracing of the voltage difference between two of the electrodes and is what is actually produced by the ECG recorder. This thesis will use the term lead to indicate the electrical cable attaching the electrodes.

### 2.3.2 Photoplethysmographic

The absorption of light of blood varies depending on the amount of oxygen it contains [118, 138]. A photoplethysmographic measurement is normally accomplished using a pulse oximeter. A pulse oximeter shines two beams of light onto a part of the body, such as a finger, earlobe or forehead, one beam is red light, and the other is infrared light. From the two light sources the pulse oximeter can calculate the oxygen saturation. However, other parts of the body will also absorb light (such as venous blood, bone, skin, muscle). To calculate the oxygen level of the arterial blood a pulse oximeter detects the slight change in the overall light absorption caused by a beat of the heart pushing arterial blood into the finger, as shown in Figure 2.5.

![Figure 2.5: Light Absorption Through the Finger](image)

The change in light absorption is very small so pulse oximeters work best when there
is a good strong blood flow in the finger (or other body part) the device is on. If the signal is too low the measured light absorption may not be reliable and lower than this the pulse oximeter will not be able to work.

At the measuring site there are constant light absorbers that are always present, such as the skin, tissue, venous blood, and the arterial blood. The variable light absorber is caused with each heart beat, the heart contracts and there is a wave of arterial blood, which increases arterial blood volume across the measuring site. This results in more light absorption during the peak of the wave. Since peaks occur with each heartbeat or pulse, the term pulse oximetry was coined. This solved many problems inherent to oximetry measurements in the past and is the method used today in conventional pulse oximetry. In practice there are complications, these include oxygen levels changing, and external noise.

Oxygenated haemoglobin absorbs more infrared light and allows more red light to pass through. Deoxygenated (or reduced) haemoglobin absorbs more red light and allows more infrared light to pass through. Red light is in the 600 to 750 nm wavelength light band. Infrared light is in the 850 to 1000 nm wavelength light band.

### 2.4 Sensor Development Platforms

This section describes a number of tools and sensor development platforms that can be used to evaluate security protocols. The simulators and emulators that can be used are also described.

#### 2.4.1 Sensor Motes

Most sensors consist of six major components, they are the:

- Processor.
- Sensor Interface.
- Memory.
- Operating System.
- Wireless Communication.
- Power Supply.

The processor executes the applications task on the sensor node. In most applications the amount of processing on a sensor node is small [111]. Some of the more popular processors are the ATmega 128L and Texas Instruments (TI) MSP430.

The ATmega 128L processor is an 8–bit microcontroller, with a 16 MHz clock. The processor can achieve up to 16 million instructions per second, and has 128 KB of flash
memory. The Atmel processor consumes 8 mW in active mode and 75 $\mu$W in sleep mode. The ATmega processor has been used on the following platforms:

- Mica motes [129].
- BTnode [23].
- Nymph [3].
- DSYS25 [17].
- Ember [43].
- Fleck [176].

The TI MSP430 processor [123] is a 16–bit microcontroller, with a 16 MHz clock. The processor has up to 120 KB of flash memory and 10 KB of RAM. The MSP430 processor requires 3mW in active mode and 15 $\mu$W in sleep mode. The MSP430 processor has been used on the following platforms: Telos [129]; Tmote Sky [92]; eyesIFxv2 [75]; Ant [85]; Pluto [117]; BSN node [182].

Wireless communication consumes the most power out of all the components in a WSN. Platforms such as the Mica2 are based on the Chipcon CC1000 chipsets. This chipset replaces the less reliable RFM TR1000. With the advent of the IEEE 802.15.4 standard for WSNs, most of the new platforms are using the Chipcon CC2420 chipset. The CC1000 delivers approximately 40 kbps, whereas, the CC2420 can reach 250 kbps.

Available memory is normally limited in sensors. For instance, the mica mote only contains 4 KB of memory. Most WSN platforms have external flash memory (EEPROM). Flash memory is used to store configuration information, and can also be used to store sensor readings. The sensor storing data allows the sensor to perform feature extraction or filtering of the sensed data, to minimise the communication overhead.

Most sensor platforms also provide analogue and digital sensor interfaces. The Atmel ATmega 128L MCU has an eight-channel 10–bit Analogue-to-Digital Converter (ADC) that can read 15 400 samples per second. A digital sensor interface is usually through serial communication, the three most common serial communication protocols that are used are I²C, SPI, and UART.

Power supply is the major determining factor on the size and lifetime of the WSN hardware. Wireless power is an encouraging technology to remove this restriction, with Berkeley’s Golem Dust designed to use an external laser beam to power up the sensor [177]. However, this method requires line of sight, and thus is unfeasible in many WSN applications. Another method is scavenging of power from movement, solar, or wind. However, the power requirements for radio transmission make batteries the main source of power.

The limited resources in WSN platforms makes conventional embedded OSes, such as Embedded Microsoft XP or Embedded Linux, unfeasible. A number of OSes have been developed for WSNs, these include: $\mu$OS; Palos; MANTIS; SOS; MagnetOS; TinyOS. The
most widely adopted OS is TinyOS [111]. TinyOS is supported on a large number of WSN platform, they include: MicaZ; DSYS25; eyesIFXv2; Fleck; Telos; BSN node. TinyOS is developed in the nesC programming language [65]. The nesC programming language is based on a C-like syntax.

To help debug and simulate WSN applications, TOSSIM has been developed. TOSSIM is a simulator for TinyOS, it allows users to compile, debug and analyse their TinyOS application on a PC. A plug-in called PowerTOSSIM is available, which simulates the power consumption of an individual sensor and the WSN.

Another useful tool is the ATEMU simulator. This tool is a virtual machine for a mica2 sensor, and displays the total number of instructions executed by an application.

### 2.4.1.1 Body Sensor Networks

Figure 2.6 shows two types of Body Sensor Network (BSN) [62]. Both types can communicate with a controller node, such as a PDA, to connect to the Internet, and thus sending data to a hospital. This allows patients to be sent home and to their work, and still enabling the medical staff to monitor them. The PDA can be replaced with a mobile phone, or other devices that can connect the sensors with the Internet [61]. The mesh network has the sensors able to directly communicate with their neighbouring sensor. The star network has the sensors communicating to a controller node.

![Different BSN Topologies](image)

*Figure 2.6: Different BSN Topologies*
2.4.2 Arduino Devices

Arduino [126] is a customisable open-source single-board microcontroller. The system is designed to make the process of using electronics in multidisciplinary projects, such as wearable devices, more accessible. The hardware consists of an Arduino board with an Atmel AVR processor and on-board input/output support. The software consists of a standard programming language compiler and the boot loader that runs on the board. The standard programming language is based on the Wiring language (syntax and libraries), similar to C++ with some modifications, and the integrated development environment is Processing-based.

An Arduino board normally consists of an 8–bit Atmel AVR microcontroller and complementary components to facilitate programming and incorporation into other circuits. An Arduino device exposes their connectors in a standard way, which allows the CPU board to be connected to interchangeable add-on modules known as shields. Most shields are individually addressable via an I²C serial bus, allowing many shields to be stacked and used in parallel. However, some shields communicate with the Arduino board directly over various pins. A vast majority of Arduino devices have used the megaAVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. Most boards include a 5 volt linear regulator and a 16 MHz crystal oscillator. An Arduino’s microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory.

2.4.3 Android Smart Phones

A smart phone [120] is a mobile phone with advanced computing capabilities. The first smart phones combined features of a PDA with a mobile phone. Later models added features found in portable media players, digital cameras, video cameras, and GPS navigation units. Most smart phones also have high-resolution touchscreens and web browsers. They also have high-speed data access, provided by WiFi and Mobile Broadband.

The capabilities and characteristics of android smart phones released in the year 2012 are listed below:

**CPU** ARM based processor, single core to quad core. The clock speed ranges from 1 GHz to 1.6 GHz.

**Storage capacity** Range from 4 GB to 64 GB of flash memory.

**RAM** Ranging from 512 MB to 2 GB of memory.

**Size** Volume ranging 86 cm$^3$ to 108 cm$^3$.

**Weight** Range from 115 g to 180 g.

**Display** 3.7 inches to 5.5 inches.
Camera 5 MPixels to 8.7 MPixels. Normally at full HD. Some contain more than one camera.

Sensors Many sensors including: 3-axis accelerometer; 3-axis gyroscope; multi-touch; magnetometer; GPS; aGPS; compass; FM radio; noise cancellation; Ambient light.

Communication mediums Include WiFi a/b/g/n, Bluetooth 4.0, NFC.

GPU Most smart phones released this year contain a GPU.

An Android smart phone contains the Android OS. The OS is open source, which Google releases under the Apache License. This open source code and permissive licensing allows the software to be freely modified by developers and researchers. Also, there is a large community of developers that extend the functionality of devices, written primarily in a customised version of the Java programming language. One of the device’s functionalities that allow to be extended is the cameras that can be found on the phone.

This format (NV21) is the standard picture format on android camera preview. YUV 4:2:0 planar images, with 8 bit Y samples, followed by interleaved V/U plane with 8 bit 2 by 2 subsampled chroma samples. The YUV model defines a colour space in terms of one luminance (Y) and two chrominance (UV) components. Y stands for the luminance component, and gives the brightness of a pixel. The U and V are the chrominance, or colour, components. A single frame in the NV21 format is shown in Figure 2.7.

![Single Frame using NV21 Format](image)

The byte stream will be Y1,. . . ,Y24,V1,. . . ,V6,U1,. . . ,U6. To calculate the colour and...
brightness of the first pixel, the values of Y1, V1, and U1 are used. The second pixel will be Y2, V1, U1 and third pixel is Y3, V2, U2. The chrominance components are reused again for Y7 and Y8. There are APIs available to obtain the width and height of a frame, and therefore, a developer can calculate which 8 bit Y value corresponds to 8 bit V and U values. Video camera technologies in smart phones enable photoplethysmographic acquisition and hence measure inter-pulse intervals, comparable to ECG and pulse oximeter readings [70].

2.5 Limitations

There are a number of issues that need to be researched in a mobile health care system to make it a reality. This thesis covers four major issues: Reliability of Security Protocols; Physiological Data as a Key; Validation of Security Assumptions; Validation of Inter-Pulse Interval.

2.5.1 Reliability of Security Protocols

One of the security vulnerabilities addressed in this thesis is the use of small keys. For performance reasons, sensors use small keys. However, small keys give an attacker a greater opportunity to compromise a sensor node. Thus, small keys will require frequent refreshing. Old keys can be used to generate new keys. But, if the old key is compromised than the new key can easily be compromised. To address this problem, an efficient and scalable mechanism of freshening the keys between the sensor nodes needs to be investigated and developed.

When sensors are deployed in a mobile open health care environment, they have a likelihood of becoming the target of adversaries. In an open environment, an individual device or an intermediate sensor node may not be completely and permanently trustworthy. Several existing protocols [45, 69] found in traditional network protocols handle the shortcomings of untrusted servers. The existing protocols have an $O(n^2)$ complexity, and are therefore not suitable to a resource constrained environment.

The KDC protocol may not be suitable for every WSN topology. For instance, it does not easily scale to a large WSN, since the non-uniform communication will focus the load onto the KDC. This may cause the battery life of the network to diminish considerably. Many of the sensor specific protocols assume that there is no heavily resourced device in close proximity. Furthermore, there is no secure mechanism available to update the keys. Keys in sensor networks are usually 64 bits in size, and they may become easily compromised. Sensors that are short-lived may not require that cryptographic keys be updated, however any sensors that exist for an extended amount of time will require updating of keys.
To make a key distribution protocol work in an environment where sensor nodes do not trust an individual base station, an authentication scheme, which can be used with limited resources and can reduce the requirement for trusting servers, needs to be found.

2.5.2 Physiological Data as a Key

Environmental data has been used as the only source of secret information to establish keys between different nodes [14–16, 130, 173]. The major benefit of using environmental data is that body sensors can use the environmental data to authenticate that another sensor is also on the same body, and not a sensor on another individual. However, researchers have cited a number of problems with this approach. The problems limit environmental data for establishing keys to only a few cases. The problems include the following:

- Every node must have the same type of sensor, and must be able to sense the same phenomenon. This may be cost prohibitive or too bulky.
- Only cryptographically strong environmental data can be used.
- BSNs can contain a heterogeneous set of sensor nodes measuring different physiological values. Therefore existing protocols, which use environmental data, cannot be used to secure communication between the nodes.
- The environmental values can become compromised, thus the new session key is also compromised.

Many of the existing key establishment protocols are designed for open environments where a sensor node can be easily compromised. There is no secure mechanism available to update the keys. Keys in sensor networks are usually 64 bits in size, and they may become easily compromised. Sensors found in BSNs may be operational for years, especially if the sensors are implanted in the body of an individual. WSN key establishment schemes, such as the random key establishment scheme, are designed for a large sensor network, and are still secure if the number of sensors compromised is below the threshold. These characteristics do not match that of the BSNs. Another drawback is that the shared keys cannot be used for entity authentication, since the same keys can be shared by more than a single pair of nodes.

An approach for a secure group key generation for off the shelf sensors with minimal memory and bandwidth resources is required. To address the problems and limitations found in body sensors, efficient and secure mechanisms of establishing keys needs to be investigated and developed.

2.5.3 Validation of Security Assumptions

Health information collected from sensors needs to be secured and in some countries (for example the USA) security is mandated. Securing a mobile health care system becomes
more challenging mainly because of the different requirements placed on various components in the system. For instance, the sensors have more severe resource constraints than the constraints found in mobile phones, cameras or desktop computers. With differences in computing power, as well as in communication costs, a range of security protocols may be required to be deployed in the entire system. For instance, a key establishment mechanism specifically suited for body sensors was proposed using physiological data [173]. However, the mobile health care system may send physiological data to medical staff or to an analytic engine [62]. The physiological data may also be sent to an actuator to release medicine into the body [62]. These may jeopardise the ability to use the physiological data in the key establishment protocol.

When the same physiological data is used for multiple purposes and/or the environment is heterogeneous and complex, it becomes important from a security or information assurance point of view to have a formal methodology to validate the system. A formal methodology is also important to insure that the information sent to medical staff and actuators to dispense medicine is accurate (secure), and the correct actions are taken. The formal methodology has a requirement that it can model the security and privacy aspects as well as the assumptions and the application correctness.

2.5.4 Validation of Inter-Pulse Interval

A key establishment mechanism specifically for body sensors was created using physiological data [173]. The sensor data that was proposed was ECG data. ECG data is the most widely used data for all security protocols that require random physiological data from the body, since it is readily available and one of the few data sources where analysis has been done to show that it is a suitable mechanism for randomness in the body [15, 172].

A problem with using ECG is that it is cumbersome to wear and use. Two points of contact on the body are required; the points of contact are connected with a wire to measure the voltage difference. Also, it is highly unlikely that there would be two devices on a body measuring ECG. Another problem with using ECG data is that smart phones do not have the capability to measure ECG unless they have another component added on that can measure ECG. For protocols that require ECG data to establish a secure protocol, this will limit the number of devices that can securely communicate with each other.

Another problem with using Inter-Pulse Interval is that there has been no study done on the randomness of a small number of heart beats. Most of the research has been investigating a sufficient number of heartbeats to create a cryptographically secure random number. The need for a detailed analysis of the characteristics and variations of such signals are needed.
2.6 Conclusion

This chapter provided a critical review of relevant research work reported in the literature. The security threats and countermeasures for a mobile health care system were described. Tools to design a complex system and some of the difficulties when verifying a complex system are also described. Examinations of available mechanisms to measure physiological data, and how to interpret the physiological data for security protocols are provided. An investigation of the main tools that are used in sensor networks when developing and evaluating sensor applications is supplied. This chapter also detailed problems and limitations of the current mobile health care systems.

The next chapter will define a research approach used to define the research questions, hypothesis and methodology. The research questions, hypothesis and methodology used in this thesis will then also be defined in the next chapter. The research approach and the problems and limitations described in this chapter will be used as a basis to define the research questions.
The best scientist is open to experience and begins with romance - the idea that anything is possible.
Ray Bradbury (1920–2012)

3

Methodology

This chapter will describe the research approach used in this thesis. Using the research approach and the problems and limitations from the previous chapter, a description of the research questions and the hypothesis for security in mobile health care systems will be provided. An explanation of the overall framework used in this research is also supplied. This chapter contains a description and evaluation of the methods, techniques and procedures used in the investigation of the overall research.

3.1 Research Approach

This research is practical rather than philosophical: its purpose is to find a solution to a problem. Theory, theoretical perspectives, theoretical structures, paradigms, methodologies and methods are regarded as research tools. In this thesis, research has been conceptualised as a pragmatic pursuit; as an exploratory activity rather than an attempt to identify truth.

The research in this thesis is positioned as positivist [37]. Positivism assumes that there is valid knowledge (truth) in scientific knowledge. Other characteristics of positivism are that there is a single correct external reality, the research focuses on generalisation and abstraction with a clear distinction between reason and feeling.

The research is also exploratory [90], this is required as from the previous chapter there is no prior knowledge of a phenomenon or a part of reality. An exploratory study
CHAPTER 3. METHODOLOGY

normally requires an understanding of past and contemporary realities, and hence usually involves a library research phase.

The methodology will be cyclic and based on design-science [5]. Design-science study emphasises both construction and improvement with an integrated outcome. A specific type of design-science is action research [90]. Action research has six properties: future oriented; collaborative; implies system development; generates theory grounded in action; agnostic; situational. Action research uses data feedback in a cyclical process.

The research in this thesis is positioned as positivist, exploratory, cyclic and a design science. The research questions, hypothesis and methodology will be based on this research approach.

3.2 Research Questions

From the problems and limitations discussed in the previous chapter, there are a number of important research questions to be addressed in the field of security for mobile health care systems. The problems and limitations include: reliability of authentication services; initialising keys; validating security assumptions; obtaining low entropy keys. The research questions identified for this research are:

**Research Question 1.** How can mobile health care systems with compromised authentication services be able to establish a secure authenticated connection?

**Sub-Question 1.1.** How can a secure and efficient key establishment protocol be developed if one or more of the key distribution nodes is compromised in a mobile health care system?

**Sub-Question 1.2.** How can secure and efficient protocols using symmetric keys be found to update keys between sensors, even if the previous session key is compromised?

**Sub-Question 1.3.** How can a set of protocols be developed to optimise different user requirements: Security; Efficiency?

**Research Question 2.** How can a family of key establishment protocols be created that use changing low entropy physiological data, found in a mobile health care system, to authenticate the nodes?

**Sub-Question 2.1.** If sensors are to be removed from and/or added to a mobile health care system, how can a secure and efficient key establishment protocol be developed to create the new session keys by using physiological data?

**Sub-Question 2.2.** How can sensed data by different nodes be used to reduce the complexity and increase the security of key generation between sensors?

**Sub-Question 2.3.** How can a set of sensors, which measure different types of sensed data from each other, make use of the sensed data to create secure new keys between two devices?

**Sub-Question 2.4.** How can security protocols be found to update keys between sensors
3.3 HYPOTHESIS

by using physiological data, even if all the previous security keys have been compromised?

**Sub-Question 2.5.** How can a set of sensors calculate a secure group key between more than two devices using sensed data?

**Research Question 3.** How can security protocol assumptions be verified in a complex mobile health care system?

**Research Question 4.** How can physiological data generate a low entropy key in a timely manner?

### 3.3 Hypothesis

The research questions are addressed mainly using the following hypothesis:

**Hypothesis 1.** New authenticated key establishment protocols, based on symmetric key KDC protocols, can be developed to handle one or more key distribution nodes becoming compromised.

The new authenticated key establishment protocols will be able to handle one or more compromised authentication services in a mobile health care systems. New secure and efficient protocols should also be able to update keys between sensors, even if the previous session key is compromised. The set of protocols should be optimised to handle different user requirements: Security; Efficiency. The security protocols analysed and developed can be used as a basis to address Research Question 1 and the sub-questions.

**Hypothesis 2.** New key establishment protocols can be developed that use changing physiological data as a low entropy key to set up initial security keys.

A family of key establishment protocols (based on protocols using non-changing low entropy keys) will be created that use changing low entropy physiological data, found in a mobile health care system, to authenticate the nodes. Secure and efficient key establishment protocols will create new session keys in a dynamic environment where sensors are frequently added and removed. The protocols should use the sensed data to reduce the complexity and increase the security of key generation between sensors? The new protocols should be able to handle a set of sensors, which measure different types of sensed data from each other. The new security protocols should retain the feature of security protocols using non-changing low entropy keys and be able to update keys between sensors, even if all the previous security keys have been compromised? A security protocol should be developed that can use sensed data to calculate a secure group key. The security protocols analysed and developed can be used as a basis to address Research Question 2 and the sub-questions.

**Hypothesis 3.** Genetic Design Methodology can be used to verify security protocol assumptions in a complex system.

Genetic Design Methodology can be used to verify requirements; this can be used as a basis to check if the security protocol assumptions can also be verified. Genetic Design
CHAPTER 3. METHODOLOGY

Methodology can be used to address Research Question 3.

**Hypothesis 4.** *A small number of Inter-Pulse Intervals can be used to generate a low entropy key.*

A statistical analysis on physiological data will be performed. Also, physiological values will be measured using Android smart phones and low-cost Arduino devices. This is used as a basis to address Research Question 4.

### 3.4 Research Methodology

The evolutionary methodology design used in this thesis has a formal origin that has various names. The *Plan, Implement, Review, Improve* PIRI cycle [103] or an Experiential Learning Cycle [128], both of which are versions of Action Research [8]. The methodology used in this thesis is based on Action Research. A diagrammatic representation of the methodology is shown in Figure 3.1. There are four major phases and the stages in each phase are given below:

- **Investigate**
  - Examine WSN Environments
  - Examine BSN Environments
  - Examine Security Protocols

- **Modify/Propose Protocols**
  - Analyse Existing Protocols
  - Modify Protocols
  - Propose Protocols

- **Implement/Evaluate**
  - Security Analysis
  - Implement
  - Performance Analysis
  - Recommend

- **Validate**
  - Model Health Care System
  - Integrate Protocols in System
  - Use RBTs to Validate System
  - Validate Physiological Values

Figure 3.1 shows the order in which each phase and stage will be performed. There are three starting stages in the *Investigate* phase. A mobile health care system has several types of networks, two of which are WSNs and BSNs. Detailed characteristics of these
environments and the types of protocols used in these environments are investigated. Also, security protocols from other environments are examined in detail. After the stages Examine WSN Environments, Examine BSN Environments and Examine Security Protocols have completed, the next stage is to Analyse Existing Protocols in the Modify/Propose Protocols phase. This phase searches for protocols that may be suitable for a WSN environment.

![Overall Methodology](image)

**Figure 3.1: Overall Methodology**

When the comparison of protocols in the Modify/Propose Protocols phase is completed, the next stage is either the Modify Protocol or the Propose Protocols stage. If suitable protocols are found in the Analyse Existing Protocols stage, then modifications may be performed on those protocols in the Modify Protocol stage, to make them more suitable for a sensor environment. However, if no protocols are found to be suitable, then new protocols will need to be developed in the Propose Protocol stage. Propose protocols can also be developed based on protocols from the Modify Protocols stage.

The Modify/Propose Protocol phase may produce a number of protocols that are suitable for a WSN and BSN environment. The next stage is the Implement/Evaluate stage, where the protocols are implemented and performance is evaluated, and recommendations are made based on security and efficiency. The Validate stage is the final phase in the methodology. In this stage, a validation is performed on the security assumptions to check that they are correct for a mobile health care system, and the number of physiological data needed to create a low entropy key.
CHAPTER 3. METHODOLOGY

3.4.1 Investigate

The investigation phase has three major steps, as shown in Figure 3.2. A mobile health care system has several types of networks. Two of the major networks that are investigated are the WSNs and BSNs. Also, key establishment protocols that are designed for all networks are examined, as well as key establishment protocols designed specifically for WSNs and BSNs.

![Diagram of Investigation Phase of the Methodology](image)

**Figure 3.2: Investigation Phase of the Methodology**

3.4.1.1 Examine WSN Environments

The investigation stage includes examining characteristics of WSNs. Some of the more important characteristics include the scale of the network, the number of nodes, the number of different tasks performed by each sensor, and the security of the environment. An example of a WSN can be a home sensor network or a surveillance sensor network. In a WSN, the scale of the network can be kilometres in size, and may exist in an open insecure environment. The number of sensors can be in the thousands, and each sensor normally senses one environmental phenomena.
3.4. RESEARCH METHODOLOGY

3.4.1.2 Examine BSN Environments

When examining BSNs, the scale of the network is either the size of a limb or the body, and the sensors exist in a secure environment. The number of sensors is normally less than ten, and each sensor may measure more than one piece of piece physiological data. Other characteristics that are important when examining environments include:

- The data rate.
- The encryption requirements of data sent by sensors.
- The authentication of messages.

Some characteristics are the same for both environments, such as:

- The available memory in the sensors.
- Communication costs.
- The available energy.

Understanding the characteristics and the efficiency requirements of a sensor environment helps develop and select suitable protocols that can be used in those environments.

3.4.1.3 Examine Security Protocols

The Investigate phase also includes examining existing security protocols, either in traditional networks, WSNs or BSNs. Some of the characteristics of the protocols that were examined are security and efficiency of protocols. When examining the efficiency of the protocols, (and evaluating the protocols for the sensor environment) the communication costs and memory footprint of the protocol are important. Other characteristics include the type of confidentiality algorithm that can be used in the protocol. Protocols that do not require a specific cryptographic cipher have an advantage since sensor environments may only require integrity algorithms.

Other characteristics that are important in determining if a protocol is suitable for a sensor environment include:

- The number of nodes in the protocol.
- Forward secrecy.
- Method of key generation.
- Key confirmation.
- The strength of any existing keys.
- The security and efficiency of updating keys.

The security of the existing protocols depends on if they can withstand protocol attacks including: eavesdropping; modification; replay; preplay; reflection; denial of service; typing; cryptanalysis; protocol interaction. Understanding the attacks, and whether these
attacks are feasible in a sensor environment, are important aspects in determining valid protocols for a sensor environment.

3.4.2 Modify/Propose Protocols

This phase contains three subsections for each of the stages in the Modify/Propose Protocols phase: Analyse Existing Protocols; Modify Protocols; Propose Protocols, as shown in Figure 3.3. Following is a brief description of each phase, solutions for the research questions are then developed in each stage.

![Diagram](image)

*Figure 3.3: Modify Propose Protocols Phase of the Methodology*

3.4.2.1 Analyse Existing Protocols

The next phase finds secure and efficient key establishment protocols. Existing protocols, found in either traditional network and sensors networks, are compared with their security and efficiency features. Each research question is addressing a different problem, and thus different characteristics become important for each question.

The Research Question 2.1 can be addressed in this phase. Existing KDC protocols found in both traditional networks in sensor networks are examined. Protocols requiring sensor nodes that store old messages to prevent replay attacks have been removed. If an optimised version of the protocol exists, it will be looked at in preference to the older protocol. The requirement of a flexible protocol for most situations helped rule out the
3.4. RESEARCH METHODOLOGY

protocols relying on timestamps, as not all sensor environments are guaranteed to have secure time synchronisation. Another requirement is to minimise the amount of communication, thus protocols requiring the least number of messages are found. Protocols are then categorised on the security characteristics, such as their method of key generation and key confirmation.

The Research Question 1.1 can be addressed in this phase. In this case a multiple server KDC protocols and the communication costs are examined.

The Research Question 2.2 can be addressed in this phase. Unlike the first two questions, this question is based on a BSN environment rather than other sensor environments. In this case password protocols are examined, since the physiological data can be cryptographically weak. The security and efficiency of the password protocols are examined in the context of the security and efficiency requirements of the BSN. The size of the RSA exponents and elliptic curve points for each of the protocols are also compared, since the sizes impact the size of the messages in the protocols. The frequency of the sensed data changing also impacts the type of protocols that can be used.

The Research Question 2.3 can be addressed in this phase. Three-party password protocols are examined as a way to solve this question. Existing three-party password protocols are examined.

The Research Question 1.2, 2.4 can be addressed in this phase. In this case, both BSNs and WSNs are considered, since both networks require an efficient mechanism to update the session keys. When looking at updating session keys, a different protocol may be used to create an initial key over the updated key. When discovering protocols, the efficiency and security of the protocols are examined. Forward secrecy is important if the initial key is small, thus an adversary will be able to compromise the initial key.

The Research Question 1.3 can be addressed in this phase. In the WSN environments, there are different security and efficiency requirements. In environments where a key can be used for multiple protocols, then some protocols are not suitable in those environments (even if they are more efficient). The same is true for the BSN environments. The environments and existing protocols are mapped depending on their security and efficiency requirements.

The Research Question 2.5 can be addressed in this phase. This question is based on a BSN environment where physiological data can be gathered rather than other sensor environments. In this case password protocols and other group protocols are examined. The security and efficiency of the password and group protocols are examined in the context of the security and efficiency requirements of the BSN. The size of the RSA exponents and elliptic curve points for each of the protocols are also compared, since the sizes impact the size of the messages in the protocols. The frequency of the sensed data changing also impacts the type of protocols that can be used.
3.4.2.2 Modify Protocols

There may be cases where the traditional protocols send extra information that is not important in a sensor environment. The protocols that have minor modifications and enhancements are labelled as Modified Protocol throughout this thesis.

The Research Question 2.1 can be addressed in this phase. The existing protocols that are chosen may then be modified so that information that is not important to the security of the keys is removed.

The Research Question 1.1 can be addressed in this phase. If an existing protocol is chosen, then it may be modified to suit a multi-tiered network environment found in home networks.

The Research Question 2.2 can be addressed in this phase. Extensions to existing protocols are examined, in the context of the sensed data frequently changing. In traditional networks, the password normally does not change between protocol runs. However, in a sensor network the sensed data, which replaces the password, may be different. The feature of the sensed data may allow for modifications to password protocols, where the protocols assume the password may stay the same between protocol runs.

The Research Question 2.3 can be addressed in this phase. Modified protocols may be created by using the same methodology as described for modifying protocols for the Research Question 2.2. The frequency of the sensed data changing (over time) may allow the modification of any existing protocols.

The Research Question 1.2, 2.4 can be addressed in this phase. When modifying an update protocol, extra information found in sensors may be used to update the keys.

The Research Question 1.3 can be addressed in this phase. The environments and modified protocols are mapped depending on their security and efficiency requirements.

The Research Question 2.5 can be addressed in this phase. Extensions to existing protocols are examined, in the context of the sensed data frequently changing. In traditional networks, the password normally does not change between protocol runs. However, in a sensor network the sensed data, which replaces the password, may be different. The feature of the sensed data may allow for modifications to group protocols that use passwords, where the protocols assume the password may stay the same between protocol runs.

3.4.2.3 Propose Protocols

There are cases where small modifications are not enough, and major changes are required to existing protocols. In these cases, the protocols have been labelled as Proposed Protocols.

The Research Question 2.1 can be addressed in this phase. Proposed protocols are created by combining characteristics from different protocols to form a new proposed
3.4. RESEARCH METHODOLOGY

protocol. When creating the proposed protocols, the security of the protocols is examined, including different attacks such as eavesdropping, modification, replay, preplay, reflection, typing, and protocol interaction.

The Research Question 1.1 can be addressed in this phase. New multiple server KDC protocols are created. Due to the communication restrictions found in sensors, the communication costs should not be over $O(n)$. When creating the proposed multiple server protocols, the requirement for the KDC servers to know the keys between each of the sensor nodes has been removed, and thus helping to limit the message sizes. Also, different sensor network environments are examined. Assumptions are made that some nodes, such as a camera, can use a different communication medium to the WSN medium. The proposed multiple server protocols can be extended to utilise this feature. The security and efficiency of the proposed protocols are examined.

The Research Question 2.2 can be addressed in this phase. New protocols can be created by using the assumption that the sensed data, which is a secret between the two nodes, may change with each run of the protocol. As with the other research questions, the security and efficiency of the proposed protocols are examined.

The Research Question 2.3 can be addressed in this phase. New three-party password protocols are created by combining three-party KDC protocols with two-party password protocols. The two-party password protocol is chosen from the research performed for the Research Question 2.2. The number of bytes sent by each of the protocols is examined before choosing the protocols. The security of the proposed protocols is examined in detail.

The Research Question 2.5 can be addressed in this phase. New protocols can be created by using the assumption that the sensed data, which is a secret between the groups of nodes, may change with each run of the protocol. Also, the protocol should be able to handle sensors added and removed from the environment. As with the other research questions, the security and efficiency of the proposed protocols are examined.

After this stage, there will be several protocols chosen. The next stage will compare the protocols; the methodology is explained in the next sub-section.

3.4.3 Implement/Evaluate

There are a number of different tests and comparisons that are performed in this stage, as shown in Figure 3.4.
3.4.3.1 Security Analysis

A review of security proofs on an existing security protocol is performed, and a review of any known attacks on an existing protocol is completed. When modifying an existing security protocol, or proposing a new protocol, a security analysis on the protocol is performed by checking different attacks such as eavesdropping, modification, replay, preplay, reflection, typing, and protocol interaction.

3.4.3.2 Implement

Instead of using simulators, a implementation and comparison is performed of the different security protocols on a Crossbow mica2 MPR2600 mote [74]. The protocols were developed using nesC and TinyOS. The experiments had one sensor attached to a workstation while the other sensors were placed stand-alone. After the running of the protocols, the elapsed time was then sent via the serial connection, to a PC running a Linux® distribution where the experiments have a Java® application reading the TinyOS packet from the serial port, and report that data to the user.
3.4.3.3 Performance Memory Analysis

By implementing the protocols using nesC and TinyOS, the memory, number of instructions executed, and amount of energy consumed, can be measured by tools such TOSSIM and ATEMU. Memory is a scarce resource in sensors, and a protocol using less memory has an advantage as it allows an application to use more memory. The memory can be calculated by the avr-size program. The combinations of `.bss`, `.text` and `.data` are examined. The `.bss` and `.data` segments use SRAM, and the combination of `.text` and `.data` segments use ROM.

The number of instructions used by each of the protocols is also important. In this case the ATEMU tool is used. The ATEMU tool calculates the number of instructions used in the mica2 mote. Finally, the memory usage of the communication can be calculated by using the TOSSIM simulator.

Measurement of an implementation of the protocols is not the only way of performing comparisons. Another analysis is to compare the number of cryptographic operations, the number of messages, and the number of bytes sent. The cryptographic operations include: encryption; decryption; MACs; pseudo-random numbers. The calculations are performed on each of the nodes, showing that some nodes require fewer operations over other nodes.

The efficiency of the protocol is also examined on different network architectures. The multiple server protocols comparisons are performed on a sensor environment where there is only one communication medium, and then the same measurements are performed on a sensor network, where there can be more than one communication medium.

The efficiency of the protocols is not the only comparison that is performed. The security of the protocol is also compared. If there is an attack, an analysis is done to show in what type of environment this attack is feasible and not feasible.

3.4.3.4 Recommend

After the comparison, a recommendation is made on which protocol to use in a particular environment. The security and efficiency requirements of a network should match the security and efficiency characteristics of the security protocol.

3.4.4 Validation

The next stage is the Validation stage, where there are a number of steps, as shown in Figure 3.5.
Validation of security protocols is an important area when checking the security correctness. The complexity of security protocols makes validation a difficult task. Computer assisted methods include model checking and theorem proving. Another form of validation is a mathematical proof. The research approach will use informal methods to evaluate the security correctness of the protocols. A review of security proofs on an existing security protocol is performed, and a review of any known attacks on an existing protocol is completed. When modifying an existing security protocol, or proposing a new protocol, an informal security analysis on the protocol is performed by checking different attacks such as:

- Eavesdropping.
- Modification.
- Replay.
- Preplay.
- Reflection.
- Typing.
- Protocol interaction.

Some of the new protocols may be based on protocols that have no formal proof, however, the existing protocols may not have any known security flaws. It is beyond the
3.5. ETHICS

scope of this thesis to find proofs for existing protocols. Instead a more practical validation step is incorporated, such as verification that the protocol can be implemented.

3.4.4.1 Model Health Care System

A mobile health care system is defined in this phase. The health care system will allow remote monitoring of patients. The system will have different types of networks, and devices can be added and removed. Some of the devices may not be physically secured, and therefore be compromised.

3.4.4.2 Integrate Protocols in System

The different types of protocols found in the Modify/Propose Protocols phase are put in the mobile health care system. The protocols range from the initial key establishment to updating keys using a KDC setup.

3.4.4.3 Use RBTs to Validate System

Genetic Design Methodology (GDM), as described in Section 2.2.2, is used to define the protocols and the mobile health care system. A test will be performed that will check if GDM can be used to verify that the protocols can be safely used in a mobile health care system.

3.4.4.4 Validate Physiological Values

The validation of physiological values will be a number of different tests. An android phone will be used to measure Inter–Pulse Intervals (IPIs), to confirm that pervasive devices such as a smart phone can be used to measure IPIs. Two mobile phones will be used; one on each hand, to confirm that two phones measuring IPIs, will measure the same values.

A database of ECG data will be used to check the variability of IPI values over a wider population of healthy people. A statistical analysis of the IPIs will be performed. The entropy of the values will be checked against the protocols that were discovered in the previous stages. If at this stage it is discovered that new protocols are needed, the next step will loop back to the Modify/Propose Protocol stage.

3.5 Ethics

One of the dilemmas of research is the philosophical ethics of the experiments on human participants. It is important that the experiments that are run are ethical. Questions should
be answered, such as is the privacy of the person guaranteed, are minors involved, are
the experiments invasive. The complete list of questions that should be checked before
running the experiments are found in Appendix A.

The ethics application number is ICT/10/11/HREC. A requirement is before data can
be gathered from an individual they will need to be given the information sheet as shown
in Appendix B. After reading the sheet they will be supplied the consent form, as shown
in Appendix C, which they will need to read and sign. After the form is signed, the data
gathering can be performed.

3.6 Conclusion

This chapter described the research questions and the hypothesis. The overall methodology
and framework that is used in this research are explained. A detailed description of the
methods, techniques and procedures used in the investigation are also given.

The methodology, techniques and procedures are used in the following chapters. Below
is a description on which phase is used in which chapter.

• Investigate
  – Examine WSN Environments Chapter 4 will go into further detail about a WSN
    environment for a mobile health care system, especially in the context of reli-
    ability.
  – Examine BSN Environments Chapter 5 will go into further detail about a BSN
    environment in the context of mobile health care systems and security.
  – Examine Security Protocols Chapter 4 and Chapter 5 will go into detail about ex-
    isting protocols either in the sensor environment and non-sensor environments.

• Modify/Propose Protocols
  – Analyse Existing Protocols Chapter 4 and Chapter 5 will compare the existing
    protocols found in sensor networks and non-sensor networks.
  – Modify Protocols Chapter 4, Chapter 5 and Chapter 6 contains the descriptions
    of the modified protocols.
  – Propose Protocols Chapter 4, Chapter 5 and Chapter 6 contains the descriptions
    of the proposed protocols.

• Implement/Evaluate
  – Security Analysis Chapter 4, Chapter 5 and Chapter 6 contains a security ana-
    lysis of the existing, modified and proposed protocols.
  – Implement Chapter 4 and Chapter 5 contains a description and the results of the
    implementation and testing of the protocols on mica2 motes.
  – Performance, Memory Analysis Chapter 4 and Chapter 5 contain performance
and memory analysis of the protocols on mica2 motes.

- **Recommend** Chapter 4 contain recommendations of which protocols to use in different types of networks.

- **Validate**
  
  - *Model Health Care System* Chapter 6 provides the detail description of the mobile health care system. The other chapters provide a brief description.
  
  - *Integrate Protocols in System* Chapter 6 contain information about when the new proposed protocols should be used in the mobile health care system.
  
  - *Use RBTs to Validate System* Chapter 6 describes the results of using RBTs in Genetic Design Methodology on verifying protocol assumptions.
  
  - *Validate Physiological Values* Chapter 6 contains the analysis of using IPIs as a suitable low entropy key. Also, a validation step is performed to check that android smart phones and low-cost Arduino devices can be used to measure IPI values to be used in the new protocols.

The next three chapters will describe detailed investigation and results of the methodology, techniques and procedures described in this chapter. Reliable security is an important component in a mobile health care system. In particular the next chapter deals with compromised authentication services in mobile health care systems.
This chapter describes the importance of reliability for security protocols in a mobile health care system. A detailed description and limitations of existing reliable key establishment protocols for sensor networks is supplied. To create a new reliable key establishment protocol a survey of key establishment protocols that use symmetric keys is completed. Several features from different protocols are closely examined to facilitate the development of a new reliable secure protocols as well as new protocols to efficiently updating keys. An analysis of the security of each of the protocols is supplied. Mobile health care systems may have different types of networks, and hence have different data rates. New reliable secure protocols are created that takes advantage of the different networks in a system. An implementation of the protocols is tested in a simulator, emulator and real hardware. An analysis of the performance results from each of the test-beds is given. A recommendation map is also given based on the network topology, security requirements, and performance consideration.

4.1 Introduction

Proposed sensor systems contain many different components and hence are inherently complex [100, 106, 185]. The mobile health care system, with both body and external sensors is an example of a complex sensor network system. The complexity of the system increases as more sensors are added to obtain more data. For instance, blood pressure increasing due to exercise is normal. However, increase in blood pressure while at rest
could mean a serious medical condition. Sensors may not just measure physiological values, but also body motions, which can lead to a number of different sensors needing to communicate with each other. As the number of heterogeneous sensors increases, so will the complexity of interactions between the sensors.

Figure 4.1 gives a diagrammatic representation of a home automation system that can be used to monitor the elderly. The diagram shows the communication of the sensors with the central home controller. The home controller is a specialised device that is situated at home and is connected to the Internet. In the case where this system is part of a health system, the home controller is a specialised device that communicates the necessary information to the hospital. Depending on the health risks and privacy concerns of the patient all of the information may not be transmitted to a hospital. For instance, cameras (home sensors) may only start recording if the body sensors detect that there may be a medical emergency, such as the patient lying horizontal in the kitchen. Surveillance software can be used to detect if the patient is cleaning the kitchen, or getting something from the ground, or there is actually an emergency. If the software does detect an emergency, the hospital staff is notified; they examine the information, and decide on the best course of action.

![Figure 4.1: Home Automation System](image-url)
In the system the messages are short-lived. If a sensor senses data, then within several minutes that information will be relayed back to the home controller. The major requirement of the message is to have information assurance with a MAC attached to each message. Privacy is also important; some sensed data (such as motion sensors) can indicate to a burglar that someone is home. Other data from sensors such as data from temperature sensor for a room is not as critical to keep private.

Motion sensor data should be encrypted and hidden for privacy reasons. However, motion sensor data from several weeks ago is not as useful to a burglar as motion sensor data at the current moment. The messages do not need to be secured for months, years, or centuries and thus small keys can be used. In a large complex system where there are many changing requirements, it is important that when a new requirement added to the system does not break another requirement. If the system is used to collect health information, then the data needs to be secured and in some countries (for example the USA) security is mandated [167]. Securing a home automation system becomes more difficult mainly because of the different requirements for various components. For instance, the sensors have dramatically more resource constraints than the constraints found in mobile phones, cameras or desktop computers. With differences in computing power, as well as differences in communication costs, different security protocols may be required throughout the entire system. Efficient key establishment mechanisms depending on different bandwidth limitations between sensors are needed.

This chapter will explain the development of efficient protocols that use the differences in computing power, as well as differences in communication costs. They address the use of small keys in their protocols. For performance reasons, sensors use small keys; however, small keys give an attacker a greater opportunity to compromise a sensor node. Thus, small keys will require frequent refreshing. Old keys can be used to generate new keys. But, if the old key is compromised, then the new key can easily be compromised. A secure, efficient and scalable mechanism to freshen the keys between the sensors nodes will be created.

The complex nature of operation of sensor network applications involves use of different communication technologies, ranging from Cellular, WLAN, WiMax, WPN, WMN, WSN. Although much work has been done on the inter-operability between these networks, the authentication of nodes and the level of security available for services in such situations are still needs innovative solutions.

The quality of security is a very important criterion of a mobile health care system. Security services should be available to facilitate the smooth operation of the system even if a component malfunctions or is compromised. As the numbers of components in a home health care system increases so will the likelihood of failure. The system should be designed so that the failure of a component does not jeopardise the security of the system.
CHAPTER 4. RELIABLE SECURITY PROTOCOLS

When WSNs are deployed in homes, they have a likelihood of becoming the target of adversaries, or devices may fail and give incorrect information. An individual server or an intermediate sensor node may not be completely and permanently trustworthy. If they are connected to the Internet, external adversaries can more easily be granted access to the devices. To make a key distribution protocol work in an environment where sensor nodes should not trust an individual base station, an authentication scheme, which can be used with limited resources and can reduce the requirement for trusting servers, needs to be found.

4.2 Notation

The notation used throughout this chapter is described in Table 4.1. The protocols using a multiplicity of servers an assumption is made that \( A \) and \( B \) do not trust any individual server. TinyOS is used as the development environment, with the following restrictions on the size of the data structures. The key size is 64 bits, the nonce size is 1 byte, the packet size is 29 bytes, and finally the location size is 2 bytes (which allows 64K of nodes in a sensor network).

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A ) and ( B )</td>
<td>The two nodes who wish to share a new session key.</td>
</tr>
<tr>
<td>( S )</td>
<td>A trusted server.</td>
</tr>
<tr>
<td>( S_i )</td>
<td>A server in a set of servers ( S_1, \ldots, S_n ), where ( n ) is the numbers of servers.</td>
</tr>
<tr>
<td>( N_A )</td>
<td>A nonce generated by ( A ).</td>
</tr>
<tr>
<td>( {M}_K )</td>
<td>Encryption of message ( M ) with key ( K ) to provide confidentiality and integrity.</td>
</tr>
<tr>
<td>( [[M]]_K )</td>
<td>Encryption of message ( M ) with key ( K ) to provide confidentiality.</td>
</tr>
<tr>
<td>( [M]_K )</td>
<td>One-way transformation of message ( M ) with key ( K ) to provide integrity.</td>
</tr>
<tr>
<td>( K_{AB} )</td>
<td>The long-term key initially shared by ( A ) and ( B ).</td>
</tr>
<tr>
<td>( K'_{AB} )</td>
<td>The value of the new session key.</td>
</tr>
<tr>
<td>( K_{AS}, K_{BS} )</td>
<td>Long-term keys initially shared by ( A ) and ( S ), and by ( B ) and ( S ) for centralised authentication server.</td>
</tr>
<tr>
<td>( K_{AS_i}, K_{BS_i} )</td>
<td>Long-term keys initially shared by ( A ) and ( S_i ), and by ( B ) and ( S_i ), for each ( i \in 1, \ldots, n ).</td>
</tr>
<tr>
<td>( X, Y )</td>
<td>The result of the concatenation of data strings ( X ) and ( Y ).</td>
</tr>
<tr>
<td>( A \rightarrow B : m )</td>
<td>Denotes that ( A ) sends a message ( m ) to ( B ).</td>
</tr>
<tr>
<td>( A \rightarrow S_i : m )</td>
<td>Denotes that ( A ) sends a message ( m ) to each server.</td>
</tr>
<tr>
<td>( S_i \rightarrow A : m )</td>
<td>Denotes that each server sends a message ( m ) to ( A ).</td>
</tr>
<tr>
<td>( X \oplus Y )</td>
<td>Exclusive-or operation with ( X ) and ( Y ).</td>
</tr>
</tbody>
</table>
4.3 Reliable Key Establishment Protocols

Many key establishment protocols assume that server is never compromised. In a sensor environment, this may not be always true. A method to alleviate the problem with a server compromising the entire system is to use multiple server key establishment protocols. Boyd and Mathuria have produced a survey of key establishment protocols using multiple servers [33] in traditional networks. In their survey, two multiple server protocols were listed: Gong’s multiple server protocol [69], and the Chen–Gollmann–Mitchell protocol [45]. However, this survey did not take into account the unique nature of a sensor environment. The main goals of using multiple servers in a sensor network are:

- Even if one or more servers become unavailable, it may be possible for the sensor nodes to establish a session key.
- Even if one or more servers are untrustworthy, the sensor nodes may still be able to establish a good key.

Protocol 4.1 Gong’s simplified multi-server protocol

| M1 | A → B : | A, B, N_A, {A, B, x_1, cc(x)}_{K_{A1}}, \ldots, {A, B, x_n, cc(x)}_{K_{A_n}} |
| M2 | B → S_i : | A, B, N_A, N_B, {A, B, x_i, cc(x)}_{K_{B_i}}, {B, A, y_i, cc(y)}_{K_{A_i}} |
| M3 | S_i → B : | {B, N_A, y_i, cc_i(y)}_{K_{A_i}}, {A, N_B, x_i, cc_i(x)}_{K_{B_i}} |
| M4 | B → A : | {B, N_A, y_1, cc_1(y)}_{K_{A_1}}, \ldots, {B, N_A, y_n, cc_n(y)}_{K_{A_n}}, \{N_A\}_{K_{AB}}, N_B |
| M5 | A → B : | \{N_B\}_{K_{AB}} |

A simplified version of Gong’s original multiple server protocol is described in [33]. The version, as shown in Protocol 4.1, is used in this thesis rather than using the more complex version. One of the main features of this protocol is that the nodes, A and B, choose the keying material while the n servers, S_1, S_2, \ldots, S_n, act as key translation centres that allow keying material from one node to be made available to the other. Initially sensor node A shares a long-term key K_{A_i} with each server S_i, and similarly sensor node B shares K_{B_i} with S_i. Sensor node A has split the key x into x_1, x_2, \ldots, x_n and sensor node B has split the key y into y_1, y_2, \ldots, y_n. The session key is defined as K_{AB} = h(x, y) where h is a one-way function. The protocol sends a total of (2n + 3) messages.

To prevent compromised servers from disrupting the protocol, sensor node A and sensor node B form a cross-checksum for all the shares. The cross-checksum for x is shown in Equation (4.1).

\[ cc(x) = (h(x_1), h(x_2), \ldots, h(x_n)) \] (4.1)

The cc_i(x) (should be equal to cc(x)) and cc_j(y) (should be equal to cc(y)) are the cross-checksums returned by server S_i. The node will give a credit point to the servers if their cross-checksum values are the same as the values obtained from the majority of servers. When all the checks are complete, sensor node B retains the value x_j with the most credit points.
The major problem with this protocol is the size of the messages. The message sizes of \(M1\) and \(M4\) in the Gong multi-server protocol are of \(O(n^2)\). Message \(M5\) is \(O(1)\), while \(M2\) and \(M3\) are of \(O(n)\). A message size of \(O(n^2)\) is not desirable in a sensor network. Another problem is that the size of the output of the one-way function will have to be reasonably large (otherwise a malicious server can quickly calculate the possible values for \(x\) and \(y\)). So for small values of \(n\), the message sizes themselves will be very large for a sensor network.

The second multiple server protocol considered is the Chen et al. multi-server protocol as shown in Protocol 4.2. One of the main features of this protocol is that the servers, rather than the sensor nodes, choose the keying material. Both nodes employ a cross-checksum to decide which servers have given valid inputs. The protocol sends a total of \((2n + 4)\) messages.

**Protocol 4.2** Chen-Gollmann-Mitchell multi-server protocol

\[
\begin{align*}
M1 & \quad A \rightarrow B : \quad A, B, N_A \\
M2 & \quad B \rightarrow S_i : \quad A, B, N_A, N_B \\
M3 & \quad S_i \rightarrow B : \quad \{B, N_A, K_i\}_{K_{A}}', \{A, N_B, K_i\}_{K_{B}}' \\
M4 & \quad B \rightarrow A : \quad \{B, N_A, K_1\}_{K_{A}}, \ldots, \{B, N_A, K_n\}_{K_{A}}', \text{cc}_B(1), \ldots, \text{cc}_B(n) \\
M5 & \quad A \rightarrow B : \quad \text{cc}_A(1), \ldots, \text{cc}_A(n), \{B, N_B, N'_A\}_{K_{AB}}' \\
M6 & \quad B \rightarrow A : \quad \{A, N'_A, N_B\}_{K_{AB}}'
\end{align*}
\]

The cross-checksum used in this protocol is different from the one used in the Gong multi-server protocol. The sensor node \(B\) calculates the \(\text{cc}_B(i)\) as shown in Equation (4.2).

\[
\text{cc}_B(i) = \{h(K_1), h(K_2), \ldots, h(K_n)\}_{K_i}, \quad \forall i \in (1, \ldots, n) \tag{4.2}
\]

To prevent sensor nodes \(A\) or \(B\) imposing the session key, the choice of \(h()\) is limited; for example, it cannot be an exclusive-or-operation. If \(B\) does not receive any message from server \(S_j\), then \(\text{cc}_B(j)\) is an error message, and \(h(K_j)\) is replaced by an error message in the calculation of the other \(\text{cc}_B(i)\) values. When sensor node \(A\) receives the checksums, it will first decrypt the values and compares the values with its own calculations of the cross-checksums. The valid \(K_i\) secrets are retained for the majority of \(i\) values and others are discarded. The session key \(K_{AB}\) is defined to be the hash of all the good \(K_i\) values concatenated, as shown in Equation (4.3).

\[
K_{AB} = h(K_1, \ldots, K_m) \tag{4.3}
\]

The messages \(M4\) and \(M5\) in the Chen et al. multi-server protocol have a computational complexity of \(O(n^2)\). While messages \(M2\) and \(M3\) have \(O(n)\), messages \(M1\) and \(M6\) have \(O(1)\) computational complexity.
4.3. RELIABLE KEY ESTABLISHMENT PROTOCOLS

This protocol encounters similar problems as the Gong multi-server protocol with regard to the size of the messages. Once again, several messages are of $O(n^2)$ in size. With the cross-checksums containing the outputs of a one-way function where the inputs are key values, once again the size of the output will need to be large. The cross-checksums in this protocol are encrypted instead of only requiring a hash algorithm. However, the Chen et al. multi-server protocol is considerably more efficient with regard to the size of the messages.

There is also another variant to this protocol, called the cascade protocol. The difference is instead of sensor $B$ making a request to each server by repeating message $M2$, the request is passed on from $S_1$ to $S_2$ and so on. The response from each server is also passed on. Finally server $S_n$ returns all the server responses to $B$, and the last three messages are the same. This may be beneficial in a sensor environment, especially if the base stations communicate through different means or media then the sensor motes. In an environment where the base stations are resource rich, this is a reasonable assumption to make. However, there is one major drawback to the protocol; it will only work if all the servers are operational. The cascading protocol version of the protocol was ruled out since a mobile health care system may have servers that were unreachable, and the cascade protocol still did not deal with the message size problems existing in the original protocol.

Another aspect of the multiple server protocols is the creation of the new key $K_{AB}$. The sensor nodes $A$ and $B$ retrieve the new key by using a secret sharing mechanism such as the one defined in [139]. Secret sharing is a mechanism allowing the owner of a secret to distribute shares amongst a group. Individual shares or a small number of shares are no help in recovering the secret. The $n$ shares are distributed, such that any set of $t$ (or more) shares is sufficient to obtain the secret. The most well-known threshold scheme uses polynomial interpolation.

When polynomial interpolation is used in cryptographic applications the field is typically $\mathbb{Z}_p$, the field of integers modulo $p$, for some prime $p$. To share a secret $s \in \mathbb{Z}_p$ in the $(t, n)$ threshold scheme, the dealer generates a polynomial of degree $(t - 1)$:

$$ f(x) = a_0 + a_1x + \cdots + a_{t-1}x^{t-1} \quad (4.4) $$

The coefficients are randomly chosen in $\mathbb{Z}_p$ except for $a_0 = s$. The shares are values of $f(x)$ with $1 \leq x \leq n$. If $t$ shares are known $s$ can be recovered. For example, if $f(1), f(2), \ldots, f(t)$ are known, then $s$ can be calculated, as shown in Equation (4.5).

$$ s = \sum_{i=1}^{t} f(i) \prod_{i < j \leq t} \frac{j}{i} \quad (4.5) $$

Given any $t$ points on the polynomial (excluding the value of 0), the value for $f(0)$ can be obtained.
The time complexity to compute $n$ shares is $O(nt)$. The time complexity to recover $a_0$ is $O(t \log_2 t)$ [69]. Having such a scheme or similar scheme in a sensor node will consume significant amount of resources in an already resource constrained environment.

The existing multiple server protocols are therefore not suitable for a sensor network environment. An ideal solution with the desired characteristics requires innovative multiple server protocols, specifically designed for the sensor environment.

### 4.4 Single Server Protocols

This section will investigate the available single server protocols with the intention of exploring the possibility of extending one of the protocols into a suitable multiple server protocol. This section will also develop two single server protocols and compared their performance, both analytically and via simulation, with the existing protocols. Of particular interest are the protocols using shared key cryptography. Boyd and Mathuria have also surveyed the protocols for authentication and key establishment [33] in traditional networks. However, the protocols have not been extensively analysed for WSN environments.

Boyd and Mathuria have listed a total of 22 server-based key establishment protocols [33]. In this analysis the number of significant protocols is reduced to a manageable size of seven. Keeping sensor environments in mind, the steps shown in Figure 4.2 have been used to filter out the desired protocols.

![Figure 4.2: Filtering Single Server Security Protocols](image-url)
4.4. SINGLE SERVER PROTOCOLS

Protocols requiring sensor nodes that store old messages to prevent replay attacks have been removed. If an optimised version of the protocol exists, it will be looked at in preference to the older protocol. The requirement of a flexible protocol for most situations helped to rule out the protocols relying on timestamps, as not all sensor environments are guaranteed to have secure time synchronisation. Another requirement is to minimise the amount of communication, this can be accomplished by removing any protocols requiring five or more messages, leaving the protocols that send only four messages, as shown in Table 4.2.

### Table 4.2: Comparison of Filtered Traditional Network Single Server Protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Control</th>
<th>Freshness</th>
<th>Confirmation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bauer–Berson–Feiertag [18]</td>
<td>S</td>
<td>A+B</td>
<td>No</td>
</tr>
<tr>
<td>Otway–Rees [124]</td>
<td>S</td>
<td>A+B</td>
<td>No</td>
</tr>
<tr>
<td>Yahalom [38]</td>
<td>S</td>
<td>A+B</td>
<td>B</td>
</tr>
<tr>
<td>Bellare–Rogaway [21]</td>
<td>S</td>
<td>A+B</td>
<td>No</td>
</tr>
<tr>
<td>AN Otway–Rees [1]</td>
<td>S</td>
<td>A+B</td>
<td>No</td>
</tr>
<tr>
<td>11770–2 Mech. 13 [87]</td>
<td>B</td>
<td>A+B</td>
<td>No</td>
</tr>
</tbody>
</table>

From the remaining seven protocols, the protocol with the most desired properties was the Boyd four-pass protocol [30]. One of the most important properties is the key confirmation property: if key confirmation is not considered important in a particular environment, then another protocol such as Bellare–Rogaway can be considered.

The Boyd four-pass protocol provides key authentication, key freshness and key confirmation in four messages, as shown in Protocol 4.3. The nodes A and B exclusively share a secret ($K_{AS}$ and $K_{BS}$ respectively) with the trusted authentication server S. By executing the protocol sensor nodes A and B intend to establish a session key $K_{AB}$.

#### Protocol 4.3 Boyd key agreement protocol

<table>
<thead>
<tr>
<th>Step</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>$A \rightarrow S$: $A, B, N_A$</td>
</tr>
<tr>
<td>M2</td>
<td>$S \rightarrow B$: ${A, B, K_S}<em>{K</em>{AS}}, {A, B, K_S}<em>{K</em>{BS}}, N_A$</td>
</tr>
<tr>
<td>M3</td>
<td>$B \rightarrow A$: ${A, B, K_S}<em>{K</em>{AS}}, [N_A]<em>{K</em>{AB}}, N_B$</td>
</tr>
<tr>
<td>M4</td>
<td>$A \rightarrow B$: $[N_B]<em>{K</em>{AB}}$</td>
</tr>
</tbody>
</table>

An attractive feature of this protocol is that $K_{AB}$ is generated using information from A, B and S, as shown in Equation (4.6). The nonces are used to guarantee the key is fresh. The nonces $N_A$ and $N_B$ can either be random numbers or a counter, although $K_S$ should be a good random number. The last two messages supply the key confirmation functionality for A and B. Also, the server should remove $K_S$ from memory. If the server itself ever becomes compromised, then it will not compromise any sensor nodes.
The requirement for $M2$ and $M3$ to have the location of $A$ and $B$ encrypted is not essential. Thus, a modified version of the Boyd four-pass protocol has been proposed. The Protocol 4.4 is the modified version without the encryption of the locations; however the locations are still used to create the MAC values.

**Protocol 4.4 Modified Boyd Key Agreement protocol**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$M1$</td>
<td>$A \rightarrow S$</td>
<td>$A, B, N_A$</td>
</tr>
<tr>
<td>$M2$</td>
<td>$S \rightarrow B$</td>
<td>$[[K_S]]<em>{K_A}, [A, B, K_S, [[K_S]]</em>{K_B}, [A, B, K_S]_{K_B}, N_A, A$</td>
</tr>
<tr>
<td>$M3$</td>
<td>$B \rightarrow A$</td>
<td>$[[K_S]]<em>{K_A}, [A, B, K_S]</em>{K_A}, [N_A]<em>{K</em>{AB}}, N_B$</td>
</tr>
<tr>
<td>$M4$</td>
<td>$A \rightarrow B$</td>
<td>$[N_B]<em>{K</em>{AB}}$</td>
</tr>
</tbody>
</table>

The security of the new protocol relies upon the unforgeability property [68] of the MAC. The security of the protocol is slightly weakened because the location names are no longer verified from both decrypting the message and the integrity checks of the MACs. When $B$ and $A$ receive their messages, the location names $B$ and $A$ can only be verified from the MACs. However, conventional security protocols err on the side of caution [95]. Most algorithms producing MACs are good enough, because the probability that the location names are not $A$ and $B$ is extremely low. The benefit to performance is considered to be worthwhile, at the cost of a minimal decrease in security. In the next subsection, an investigation is performed of the scalability of the Modified Boyd Key Agreement protocol.

### 4.4.1 Scalability Improvements when Updating Keys

The Modified Boyd Key Agreement protocol will only need to be run once between $A$ and $B$. The sensor nodes can cache $K_S$ and instead of contacting the server again, they can then use a different protocol to establish a new key. Upon further investigation, it was apparent that the Bellare–Rogaway MAP1 protocol [20], a provably secure entity authentication protocol, can be used to produce a new session key.

The MAP1 protocol, as shown in Protocol 4.5, provides mutual authentication between $A$ and $B$. If $K_{AB}$ is calculated using Equation (4.6), then this protocol becomes a key establishment protocol. The new $K_{AB}$ key is guaranteed to be fresh, since $N_A$ and $N_B$ are used to create the new key. Key confirmation for both sensor nodes is another feature of this protocol. By creating a new $K_{AB}$ this section has transformed an entity authentication protocol into a key establishment protocol.

One of the properties of the MAP1 protocol is that encryption is not used to establish the key. There are situations where wireless sensor environments do not need to support encryption, and may only need integrity checking. However, most key establishment proto-
4.4. SINGLE SERVER PROTOCOLS

Protocol 4.5 Bellare–Rogaway MAP1 protocol

| M1  | $A \rightarrow B : A, N_A$ |
| M2  | $B \rightarrow A : N_B, [B, A, N_A, N_B]_{K_{sa}}$ |
| M3  | $A \rightarrow B : [A, N_B]_{K_{sb}}$ |

...cols, where there is a trusted server involved, require some form of encryption. The reason for the integrity checking is that Bellare–Rogaway MAP1 protocol has key confirmation for both the sensor nodes. Key confirmation functionality can be removed if there is the removal of message $M_3$, and the removal of the MAC from $M_2$.

4.4.2 Removing Encryption Requirements from the Boyd Protocol

Janson and Tsudik developed an authenticated key distribution that did not require traditional encryption when establishing a new session key [89]. This research will extend their technique to remove the need for encryption in the proposed protocol. In proposed No Encryption Key Agreement (NEKA) Protocol as defined in Protocol 4.6, the following constructs are used:

$$
\begin{align*}
\Theta_A &= [A, B, K_S]_{K_{AS}} \\
\Phi_A &= [[\Theta_A]]_{K_{AS}} \\
\Theta_B &= [A, B, K_S]_{K_{BS}} \\
\Phi_B &= [[\Theta_B]]_{K_{BS}}
\end{align*}
$$

(4.7)

The $\Phi$ values can either be created from an encryption algorithm or from a MAC. In cases like CBC–MAC, where the algorithm uses an underlying encryption algorithm, it may be more efficient to use encryption. In other cases where there is no encryption algorithm available (for instance there is only HMAC–MD5, or hardware support) then the $\Phi$ values can be created by using the MAC. The size of $\Phi$ should be equal to or greater than the size of the keys. By default TinySEC has a 32 bit MAC; however, the CBC–MAC can create a variable size MAC. Another adaption was to split the message $M_2$, in Protocol 4.6, allowing the decrease of the size of message $M_3$, thus minimising the amount of data sent by sensor node $B$. However, if desired, messages $M_2$ and $M_2'$ can be recombined into a single message and then message $M_3$ will need to increase in size again. Also, if key confirmation functionality is not required, then messages $M_3$ and $M_4$ can be removed.

4.4.2.1 Security Analysis of the NEKA Protocol

In the NEKA protocol there are two messages that contain the $\Phi$ variable; the message $M_2$ — server $S$ sending to node $B$, and the message $M_2'$ — server $S$ sending to node $A$. Neither $A$ nor $B$ ever send out the $\Phi$ value. A possible attack on the proposed protocol is for an
adversary to try and obtain the \( \Phi \) values by interrogating \( S \). If an adversary pretends to be \( A \), it does not matter what locations and nonce gets passed to \( S \), because \( S \) should produce a new \( K_S \), and therefore a new \( \Phi \) and a new \( \Phi \oplus K_S \).

As shown in Equation (4.8), if two exclusive-ors produce the same value, and the keys are different, then the \( \Phi \) variable will have to be different.

\[
\Phi \oplus K_S = \Phi' \oplus K'_S \tag{4.8}
\]

If the \( \Phi \) values are the same, as shown in Equation (4.9), then no extra information about \( K_S \) can be obtained, as long as a strong MAC is used. It is assumed that the adversary does not know the long term key \( K_{AS} \).

\[
[A,B,K_S]_{K_{AS}} = [A,B,K'_S]_{K_{AS}} \tag{4.9}
\]

Since \( B \) does not initiate the protocol, it has no input into the creation of \( \Phi \). So an adversary who pretended to be \( B \) has less input in the value of \( \Phi \) than if they pretended to be \( A \). The integrity of the key is also assured since key modification requires simultaneous modification of \( \Theta \) as well as \( \Phi \oplus K_S \).

This of course will fail if either \( A \) or \( B \) become compromised. Key establishment protocols between \( A \) and \( B \) cannot detect if either one of the sensor nodes is compromised. Communication between the nodes will need to be analysed to detect if any false data has been sent [132, 187]. If the server becomes compromised, the key \( K_S \) may also become compromised. A possible solution to this is to use multiple servers to create the key, so that even if one or more servers become compromised it will not affect the security of \( K_S \).

The key \( K_S \) can be used in the future to create or renew a session key between \( A \) and \( B \). However, that relies on the assumption that the key \( K_S \) has not been compromised. Since \( K_S \) is never used as a session key or used to encrypt any plaintext, the keys \( K_{AS} \) and \( K_{BS} \) should be compromised before \( K_S \). The sensor nodes should regularly refresh \( K_{AS}, K_{BS} \) and \( K_S \) with server \( S \).

Another problem that should not be discounted is that there may be some environments where \( K_S \) is more likely to be compromised before \( K_{AS} \) and \( K_{BS} \). In this case, \( K_S \) should be refreshed more often than \( K_{AS} \) and \( K_{BS} \). The natural thought will be to run the proposed protocol from the start. However, there is an attack where an adversary can replay a
portion of a previous message, so that the new key is still \( K_S \). Note that there is no way
to guarantee that the encryption of \( A, B, K_S \) is an old value or a new value. This type of
attack can be classified as a replay attack. If the keys between the nodes and the KDCs are
updated more frequently than \( K_S \), then the proposed protocols can be run from the start
without fear of a replay attack. The different keys between the nodes and the KDCs stop
any adversary performing a replay attack, as shown in Attack 4.7.

**Protocol 4.7** Attack on Boyd key agreement protocol

\[
M_1 \quad A \rightarrow S : \quad A, B, N_A
\]

\[
M_2 \quad S \rightarrow B : \quad \{A, B, K_S\}_{K_{AB}}, \{A, B, K_S\}_{K_{BS}}, N_A
\]

\[
M_3 \quad B \rightarrow A : \quad \{A, B, K_S\}_{K_{AB}}, [N_A]_{K_{AB}}, N_B
\]

\[
M_4 \quad A \rightarrow B : \quad [N_B]_{K_{AB}}
\]

Boyd’s defence against the replay attack relies on a revocation list being available to all
parties.

### 4.4.3 Five-Pass KDC protocol

A defence to the NEKA protocol is to have a revocation list but having a revocation list
infrastructure may not be feasible in a low bandwidth, low energy, and low computational
environment. A new protocol has been created as a solution, as described in the Five-
Pass KDC (FPKDC) protocol, shown in Protocol 4.8. The fix includes the addition of a
short extra message at the start of the protocol, and having the nonces as part of the
encrypted messages. The construction of the \( \Theta \) value now includes the nonces, as shown
in Equation (4.10). Both nonces are added for the strongest security [49].

\[
\Theta_A = [A, B, N_A, N_B, K_S]_{K_{AB}}
\]

\[
\Theta_B = [A, B, N_A, N_B, K_S]_{K_{BS}}
\]

(4.10)

**Protocol 4.8** FPKDC — Five-pass KDC protocol

\[
M_1 \quad A \rightarrow B : \quad A, N_A
\]

\[
M_2 \quad B \rightarrow S : \quad A, B, N_A, N_B
\]

\[
M_3 \quad S \rightarrow B : \quad \Theta_B, \Phi_B \oplus K_S
\]

\[
M_3' \quad S \rightarrow A : \quad \Theta_A, \Phi_A \oplus K_S, N_B
\]

\[
M_4 \quad B \rightarrow A : \quad [N_A]_{K_{AB}}
\]

\[
M_5 \quad A \rightarrow B : \quad [N_B]_{K_{AB}}
\]
4.4.4 Comparison with Existing KDC Mechanisms for Sensors

This section will compare the proposed protocols with existing KDC WSN protocols, such as SPINS [127] and PIKE [42]. Other KDC WSN protocols, such as TUTWSN KDC protocol [72], do not meet the criteria of having less than five messages and are therefore not considered. The SPINS protocol is described in Protocol 4.9. It contains four messages, with S (the KDC) sending two of those messages. The KDC generates $K_{AB}$ and sends the key to both $A$ and $B$. The nonces $N_A$ and $N_B$ are used to stop any replay attacks. Neither $A$ nor $B$ confirms that the other node has the correct key. Whereas, the NEKA protocol has key confirmation for both $A$ and $B$.

Protocol 4.9 SPINS key agreement protocol

| M1 | $A \rightarrow B$ : $A, N_A$ |
| M2 | $B \rightarrow S$ : $N_A, N_B, A, B, [N_A, N_B, A, B]_{K_{AS}}$ |
| M3 | $S \rightarrow A$ : $[[K_{AB}]]_{K_{AS}}, [N_A, B, [[K_{AB}]]_{K_{AS}}]_{K_{AS}}$ |
| M3 | $S \rightarrow B$ : $[[K_{AB}]]_{K_{BS}}, [N_A, B, [[K_{AB}]]_{K_{BS}}]_{K_{BS}}$ |

Protocol 4.10 is the description of the PIKE protocol. This protocol contains three messages, with each node sending only one message. Sensor node $A$ generates $K_{AB}$, which sends the key to $S$, which in turn sends it to $B$. Sensor node $A$ can confirm that $B$ does have $K_{AB}$, however, sensor node $B$ cannot confirm that $A$ has $K_{AB}$.

Protocol 4.10 PIKE key agreement protocol

| M1 | $A \rightarrow S$ : $\{A, B, K_{AB}\}_{K_{AS}}$ |
| M2 | $S \rightarrow B$ : $\{A, B, K_{AB}\}_{K_{BS}}$ |
| M3 | $B \rightarrow A$ : $\{A, B, N_B\}_{K_{AB}}$ |

The key agreement protocol described in PIKE suffers from a replay attack. When $A$ needs to refresh $K_{AB}$, then an adversary $C$ can intercept or even initiate the protocol with $S$ and $B$, as shown in Protocol 4.11. This protocol is vulnerable to the replay attack since the messages $M1$ and $M2$ do not contain any nonces or timestamps to indicate that they are new messages. It is further compounded by the fact that $B$ is not being able to confirm that $A$ has the same key as $B$ does. Another problem with the PIKE protocol is the lack of session identifiers. This has the effect of only having one running instance of the protocol at a time. The other protocols, described in this chapter, have nonces, which can be used as session identifiers. Hence, they do not suffer from this issue. Because of the major limitations of the KDC protocol described in PIKE, this research will not investigate the security limitations any further.

Since the SPINS protocol does not have key confirmation, when a comparison is performed of the protocols, the key confirmation messages will be removed from the other protocols. In the NEKA protocol the messages $M3$ and $M4$ are removed, and messages $M4$ and $M5$ are removed from the FPKDC protocol. In Bellare–Rogaway MAP1 protocol
4.4. SINGLE SERVER PROTOCOLS

**Protocol 4.11** Attack on the PIKE key agreement protocol

<p>| | | | | | | | | | | | | |</p>
<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>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>S</td>
<td>{A, B, K_{AB}}_{K_A} &amp;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M2</td>
<td>S</td>
<td>{A, B, K_{AB}}_{K_B} &amp;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M3</td>
<td>B</td>
<td>{A, B, N_B}<em>{K</em>{AB}} &amp;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Protocol 4.5) the message M3 can be removed. The message M2 will need to be modified to remove the key confirmation segment \([B, A, N_A, N_B]\) in the message.

The MAP1 protocol sends the least number of messages. The MAP1 protocol is the best performing protocol; however, it can only be used if a \(K_S\) exists to refresh the keys between \(A\) and \(B\). The number of encryption and decryption operations is the same for both SPINS and the proposed protocols. However, the proposed protocols do have the advantage of removing any need for encryption. The number of MACs calculated by SPINS is better than the number of MACs calculated by the proposed protocols. However, sensor node \(B\) has the same number of MAC calculations. The proposed protocols and the MAP1 protocol have an extra MAC calculation since \(K_{AB}\) is calculated using Equation (4.6). Finally, each of the protocols requires a good random number to be calculated to generate either \(K_{AB}\) or \(K_S\), except in the MAP1 protocol. The table shows little difference between SPINS, and the proposed protocols. However, Table 4.3 shows that the MAP1 protocol is the most efficient protocol. A useful aspect in the proposed protocols is that the key \(K_{AB}\) can be refreshed using the MAP1 protocol.

**Table 4.3: Performance Comparison of Single Server Protocols**

<table>
<thead>
<tr>
<th></th>
<th>SPINS</th>
<th></th>
<th>PIKE</th>
<th></th>
<th>NEKA</th>
<th></th>
<th>MAP1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>S</td>
<td>A</td>
<td>B</td>
<td>S</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>messages sent</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>messages received</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>en(de)cryption</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MACs</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>PRFs</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bits sent</td>
<td>24</td>
<td>0</td>
<td>192</td>
<td>128</td>
<td>0</td>
<td>128</td>
<td>40</td>
<td>0</td>
</tr>
</tbody>
</table>

The comparison is not useful when examining the proposed protocols over multiple iterations. Table 4.4 is a comparison of the SPINS protocol, NEKA protocol, FPKDC protocol when keys are updated. The \(m\) value is the number of iterations of the protocol. Since the proposed protocols are built on top of the Boyd protocol and the Modified Boyd key agreement protocol, as shown in Protocol 4.4, and there are significant advantages in using the proposed protocols, the Boyd protocol and the Modified Boyd key agreement protocol will not be shown in the table. Also, the comparisons, shown in the table, do not contain the performance impact of key confirmation between \(A\) and \(B\). Another assumption is that the \(\Phi\) attribute in the NEKA protocol is calculated via an encryption algorithm, rather...
than a MAC. If the proposed protocols were using a MAC, then it would be necessary to move the encryption values down to the MAC row. The nonces are simple counters, and the random numbers shown in this table is for the creation of any new keys. A sensor node sending data is a very expensive operation: each bit transmitted consumes the same amount of energy as thousands of CPU instructions, and if the distance between the nodes increases, then so will the amount of energy required. A useful aspect in the proposed protocols is that the key $K_{AB}$ can be refreshed using the MAP1 protocol.

Table 4.4: Performance Comparison of Single Server Protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>SPINS</th>
<th>NEKA</th>
<th>FPKDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messages sent</td>
<td>$4m$</td>
<td>$2m + 1$</td>
<td>$2m + 2$</td>
</tr>
<tr>
<td>Encryption</td>
<td>$4m$</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>MACs</td>
<td>$6m$</td>
<td>$2m + 4$</td>
<td>$2m + 4$</td>
</tr>
<tr>
<td>PRFs</td>
<td>$m$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Bytes sent</td>
<td>$37m$</td>
<td>$4m + 28$</td>
<td>$4m + 32$</td>
</tr>
</tbody>
</table>

The next step is to compare the number of cryptographic operations when performing multiple updates of key $K_{AB}$. For simplicity an assumption is made that each operation costs the same amount. In the SPINS protocol there is $11m$ operations, where $m$ is the number of iterations of the protocol. The number of cryptographic operations performed by the proposed protocol is $(2m + 9)$. If key confirmation is also included, then the proposed protocols will have $(6m + 9)$ operations. The proposed protocols have slightly more operations than the SPINS protocol. However, after only one key refresh, the proposed protocols perform fewer operations than if using the SPINS protocol.

Figure 4.3 is a graph showing the number of cryptographic operations for each protocol. To simplify the analysis an assumption is made that each operation costs the same amount. The graph looks at the cost of each protocol over a number of iterations. The NEKA protocol and Protocol 4.5 are combined. However, the NEKA protocol and PIKE protocol have not been modified to remove key confirmation. The NEKA (No Confirm) protocol is the NEKA protocol without key confirmation. As you can see from the graph, the initial cost of the proposed protocol is greater but if the keys need to be refreshed, then the proposed protocol comes down in cost very quickly. The proposed protocol has less cryptographic operations over the PIKE protocol after only two iterations. The SPINS protocol cost is greater than the proposed protocol without key confirmation on the second iteration.

The next comparison is the number of messages sent by each protocol. The SPINS protocol has a total of $4m$ messages, where $m$ is the number of iterations of the protocol. The number of messages for the NEKA protocol with key confirmation is $(3m + 2)$ and without key confirmation the number of messages is $(2m + 1)$. Even with the higher number of messages found in key confirmation the protocol has fewer messages sent after the second iteration. The number of messages for the FPKDC protocol with key confirmation...
4.4. SINGLE SERVER PROTOCOLS

is \((3m + 3)\) and the number of messages without key confirmation is \((2m + 2)\). Once again, even with key confirmation the proposed protocol has fewer messages sent after only the second iteration. Figure 4.4 is a graph showing the number of messages sent by each protocol. Once again, this chapter has a proposed protocol with key confirmation as well as proposing a protocol without key confirmation. The proposed protocol has fewer messages sent after two iterations over the SPINS protocol. The proposed protocol without key confirmation has fewer messages sent on the second iteration compared with the PIKE protocol. The difference between the number of messages sent by the proposed protocol with key confirmation over the PIKE protocol stays constant.

Figure 4.5 is a graph showing the number of bits sent by each protocol. Even though the PIKE protocol had the least number of messages, it did send the most number of bits compared with the proposed protocol, and the SPINS protocol. The number of bits sent on the first iteration by the proposed protocol, and the SPINS protocol is almost identical after the first iteration. However, after the second iteration there is a dramatic difference between the proposed protocol and the SPINS protocol. If key confirmation was removed from the proposed protocol, then there would be another dramatic drop in the proposed protocol. The number of messages sent is not a true indication of the performance of the protocol, since the number of bytes sent is not considered. When investigating the number of bytes sent, it was found that the SPINS protocol has a total of \(37m\) bytes, where \(m\) is the number of iterations of the protocol. The number of bytes for the NEKA protocol with key confirmation is \((12m + 29)\) and number of bytes without key confirmation is \((4m + 28)\).
The number of bytes sent by the proposed protocol with key confirmation is significantly less on the second iteration. If there was no key confirmation, then the proposed protocol will have less bytes sent after the first run. The same test was performed for the FPKDC protocol, and similar results were obtained. The number of bytes for FPKDC protocol with key confirmation is \((12m + 32)\) and the number of bytes without key confirmation is \((4m + 32)\).

### 4.4.5 Simulation Results

This section uses the values provided by the TOSSIM simulator (a part of the TinyOS installation) to obtain an indication of the power consumption when sending a message. In the calculations, shown in this section, it was not taken into account any collision avoidance times. On the mica2 mote, the cost of sending a 0 byte message is 9.9 microjoules, the cost of sending an 8 byte message is 21.1 microjoules, and a 16 bytes message is 32.4 microjoules. There is a substantial start-up cost for each message sent, and then there is an added cost for every bit that is sent.

A memory comparison for the application, which was created for the experiment, was performed in a TinyOS environment. A comparison is performed between CBC–MAC using SKIPJACK and RC5, and HMAC–MD5 on the application, as shown in Table 4.5. The ROM memory is greater for HMAC–MD5 than it is for CBC–MAC. The amount of RAM used is less for HMAC–MD5.
4.4. SINGLE SERVER PROTOCOLS

![Graph showing the number of bits per iteration for different protocols.

Figure 4.5: Number of Bits per Iteration

<table>
<thead>
<tr>
<th>Memory</th>
<th>CBC–MAC (SKIPJACK)</th>
<th>CBC–MAC (RC5)</th>
<th>HMAC MD5</th>
<th>No Encrypt</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
<td>13 884</td>
<td>13 290</td>
<td>24 534</td>
<td>11 532</td>
</tr>
<tr>
<td>RAM</td>
<td>617</td>
<td>617</td>
<td>592</td>
<td>528</td>
</tr>
<tr>
<td>.data</td>
<td>80</td>
<td>80</td>
<td>144</td>
<td>80</td>
</tr>
<tr>
<td>.bss</td>
<td>537</td>
<td>537</td>
<td>448</td>
<td>488</td>
</tr>
<tr>
<td>.text</td>
<td>13 804</td>
<td>13 210</td>
<td>24 390</td>
<td>11 452</td>
</tr>
</tbody>
</table>
The combination of .bss and .data segments use SRAM, and the combination of .text and .data segments use ROM. The values in Table 4.5 indicate that HMAC–MD5 uses less RAM and more ROM than CBC–MAC. The .text contains the machine instructions for the application. The .bss contains uninitialised global or static variables, and the .data section contains the initialised static variables.

On further investigation the HMAC code itself only adds 176 bytes of .text and has no impact on the .data segment. The major memory cost in HMAC–MD5 is the cost of MD5; it uses a total of 12826 bytes. Further research into reducing the size of MD5, or investigating other one-way functions could lead to more memory efficient implementations. For instance, the .text section of HMAC–MD5 can be decreased if the common transformation macros are re-factored into functions.

It was found that the TOSSIM simulator cannot be used for obtaining very accurate results when comparing the different MAC algorithms. Instead, ATEMU was used to show the number of cycles performed by the different MAC implementations, as shown in Figure 4.6. From the initial implementations, the size of the digest does not affect the number of cycles by a significant amount. It was noticed that HMAC-MD5 is significantly larger than both SKIPJACK and RC5, and RC5 performs better than SKIPJACK. When the digest size is increased from four bytes to 8 bytes, there was no significant difference in the number of cycles. It should be noted that this is not an extensive survey of MAC functions. For instance, there is no requirement for the underlying hash function used by HMAC to be collision free (it needs to be a one-way function), which greatly increases the number of MAC functions that can be used. However, even with a naive approach to implementing MD5 the newly created protocols are created so that it is possible to use HMAC–MD5 when no encryption algorithms are available.

After a detailed analysis of the proposed protocols and the comparison with the existing protocols, a conclusion that can be derived is that the large number of features (such as key confirmation, no need for large encryption algorithms, scalability, key freshness) found in the NEKA protocol and FPKDC protocol make them ideal candidates when creating a new multiple server protocol.

### 4.5 Reliable Security Protocols for Sensors

The single server protocols assume that server $S$ is never compromised. In a sensor environment, this may not be always true. A solution is to use multiple authentication server protocols, specifically designed for the sensor environment. This research has created three new multiple server protocols specifically for the sensor environment. The three new proposed Multiple Server Protocols (MSP) will be named MSP1, MSP2, and MSP3. The multiple authentication server protocols developed are based on the concept of the Boyd four-pass protocol [33]. These will maintain similar security characteristics as that of
4.5. RELIABLE SECURITY PROTOCOLS FOR SENSORS

the centralised authentication protocol, as shown in the Boyd four-pass protocol. Because of the severe resource constraints which exist in sensor nodes, a multiple authentication server protocol should have low computational complexity in both time and space. A multiple server authentication protocol in a sensor environment should have the following characteristics:

- Small computation overhead.
- Minimal number of messages.
- Sensor nodes should not be relied upon to generate good randomness.
- The new key should be fresh.
- There should be key confirmation by both nodes.

The next several sections will go into detail about the three different sensor-based multiple server protocols.

4.5.1 Multiple Server Protocol 1

The MSP1 is an efficient multiple server protocol, which specifies \( n \) servers. The protocol has the following message flows. The sensor node \( A \) sends the first message, \( A,B,N_A \), to each of the servers. Each server sends their message to both sensor nodes \( A \) and \( B \). Sensor node \( B \) sends \( N_B \), the keying data, and the cross-checksums created by \( B \). It is important to note at this stage that \( K_S \) is unknown, so unlike the original protocol, \( B \) is not able to send \([N_A]_{K_{AB}}\). When sensor node \( A \) receives the next message, it will calculate its own cross
checksums, and compare them against the cross-checksums created by $B$. At this stage, the keys $K_S$ and $K_{AB}$ are created. Sensor node $A$ sends its cross-checksums to $B$, so $B$ can create $K_{AB}$. The final message completes the key confirmation between $A$ and $B$, as shown in the proposed Protocol 4.12.

The MSP1 provides key authentication, key freshness and key confirmation, using multiple authentication servers. In the MSP1 protocol, the following constructs are used:

\[
\begin{align*}
\Theta_{Ai} &= [A, B, K_i]_{K_{ASi}} \\
\Phi_{Ai} &= [[[\Theta_{Ai}]]]_{K_{ASi}} \\
\Theta_{Bi} &= [A, B, K_i]_{K_{BSi}} \\
\Phi_{Bi} &= [[[\Theta_{Bi}]]]_{K_{BSi}}
\end{align*}
\] (4.11)

**Protocol 4.12** MSP1 — Multiple Server Protocol 1

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>$A \rightarrow S_i$</td>
<td>$A, B, N_A$</td>
</tr>
<tr>
<td>M2</td>
<td>$S_i \rightarrow B$</td>
<td>$N_A, A, S_i, \Theta_{Bi}, \Phi_{Bi} \oplus K_i$</td>
</tr>
<tr>
<td>M2'</td>
<td>$S_i \rightarrow A$</td>
<td>$S_i, \Theta_{Ai}, \Phi_{Ai} \oplus K_i$</td>
</tr>
<tr>
<td>M3</td>
<td>$B \rightarrow A$</td>
<td>$cc_B(1), \ldots, cc_B(n), N_B$</td>
</tr>
<tr>
<td>M4</td>
<td>$A \rightarrow B$</td>
<td>$cc_A(1), \ldots, cc_A(n), [N_B]<em>{K</em>{AB}}$</td>
</tr>
<tr>
<td>M5</td>
<td>$B \rightarrow A$</td>
<td>$[N_A]<em>{K</em>{AB}}$</td>
</tr>
</tbody>
</table>

Both of the sensor nodes and the servers contribute to the key value. The values $N_A$ and $N_B$ are generated by $A$ and $B$ respectively as input to the MAC function, that determines the session key. The key used with the MAC function is generated by the servers. Both $A$ and $B$ compute the session key as $K_{AB} = [N_A, N_B]_{K_S}$. The nodes should have a minimum number of servers returning valid results before confirming that the key is valid. Node $B$ will calculate $cc_B(i) \forall i \in 1, \ldots, n$.

\[
cc_B(i) = \begin{cases} 
[K_i]_{K_i} & \text{IF VALID}, \\
EM & \text{OTHERWISE}
\end{cases}
\] (4.12)

Where $EM$ is an error message; an example will be the value zero. There is a remote chance a valid case may be zero. If the valid value is zero, the server needs to be considered a compromised server (even though it is not a malicious server).

Node $A$ will calculate $cc_A(i)$, and compare it with $cc_B(i)$. If they are the same, then the server $S_i$ is valid. Below is a way the nodes compare the cross checksum for $cc_A(i)$ and $cc_B(i)$.

\[
cc_A(i) = \begin{cases} 
cc_B(i) = [K_i]_{K_i} & \text{IF VALID}, \\
EM & \text{OTHERWISE}
\end{cases}
\] (4.13)
4.5. RELIABLE SECURITY PROTOCOLS FOR SENSORS

After the comparison of the entire cross checksums, a set of valid keys $V_1, \ldots, V_m$ should remain. The creation of $K_S$ is defined as follows.

$$K_S = V_1 \oplus \ldots \oplus V_m$$  \hspace{1cm} (4.14)

Where $V_i$ is the $i^{th}$ valid key given by a server, and $m$ is the total number of valid servers $t \leq m \leq n$, where $t$ is the minimal number of trusted servers. However, unlike the existing multiple server protocols, the trusted servers will not be able to calculate $K_S$. The calculated $cc_A(i)$ values are returned to $B$, where $B$ performs similar checks as $A$ and calculates $K_S$.

4.5.1.1 Analysis of Multiple Server Protocol 1

The MSP1 has a number of advantages, one of which is that the nodes do not need good random number generators to create the nonces. The nodes could even safely use a counter for their nonce values. Another advantage is that if a server or a number of servers are unavailable, the authentication service itself still exists through the other servers. The servers and the sensor nodes have different keys; even if one or more servers become compromised, the authentication service or the security of the system is not compromised.

The protocol only encrypts random information. If the encryption cipher uses an IV value (such as RC5 and SKIPJACK currently used in TinyOS [102]) then a constant IV value can be used. However, the constant IV value chosen for MSP1 protocol must only be used to encrypt the random data and should never be used to encrypt other information. Also, a wide variation of different ciphers can safely be used.

Some MACs have vulnerabilities when the message sizes are variable. All of the message sizes are of constant value, allowing the system to safely use a wider range of MACs than previously available. The size of the MACs can be lower than that of conventional protocols. The integrity checking is performed by the sensor nodes. If $x$ is the size of the MAC in bits, then an adversary has 1 in $2^x$ chance in blindly forging a valid MAC for a particular message. The adversary should be able to succeed in $2^{x-1}$ tries. Because of the low bandwidth of sensor nodes, a 4 byte MAC, requiring $2^{31}$ packets, will take years to complete. If an adversary did attempt this attack, the sensor node would be non-functional within that period. In addition, an adversary will need to forge $2t$ MACs; $t$ MACs to $A$ and $t$ MACs to $B$, and stop traffic from the other base stations before they can determine the value of $K_{AB}$.

In order to study the performance impacts of each of the multiple server protocols, the following symbols are first defined. The size of location indicator is $a_0$, the nonce size is $a_1$, the key size is $a_2$, the hash size is $a_3$, and the number of servers is $n$. The size of the packets sent on the sensor network is described as $a_4$. The following equations are used to...
define how many bytes are sent for each message: $M_1 = 2a_0 + a_1$, $M_2 = 2a_0 + a_1 + a_2 + a_3$, $M_2' = a_0 + a_2 + a_3$, $M_3 = a_1 + na_3$, $M_4 = (n+1)a_3$, and $M_5 = a_3$. However, there are $n$ messages of type $M_1$, $M_2$ and $M_2'$.

The following equations are used to calculate the number of packets sent out.

Equation (4.15) specifies the number of packets sent by sensor node $A$. In a typical scenario, the value of $a_0$ is 2 bytes (the network will be able to handle 64 K nodes). The nonce value is 1 byte, and the packet size is 29 bytes as is defined in TinyOS. The number of packets sent is dependent on how many servers are available.

$$M_1 = n \left\lfloor \frac{2a_0 + a_1}{a_4} \right\rfloor$$ (4.15)

Equation (4.16) specifies the number of packets received by sensor node $B$. The key size value is normally 8 bytes, and the hash value is 4 bytes.

$$M_2 = n \left\lfloor \frac{2a_0 + a_1 + a_2 + a_3}{a_4} \right\rfloor$$ (4.16)

$$M_2' = n \left\lfloor \frac{a_0 + a_2 + a_3}{a_4} \right\rfloor$$ (4.17)

Equation (4.18) specifies the number of packets sent by $B$ and received by $A$. In this case separate messages from each server have been concatenated together. This still leaves a large message which will need to be split up into several packets.

$$M_3 = \left\lfloor \frac{a_1 + na_3}{a_4} \right\rfloor$$ (4.18)

Equation (4.19) specifies the number of packets sent by $A$ and received by $B$. The cross-checksums themselves are hash values, and do not put a large overhead on communication costs.

$$M_4 = \left\lfloor \frac{(n+1)a_3}{a_4} \right\rfloor$$ (4.19)

Equation (4.20) is one of the smallest messages, where the packet overhead will be more the the payload.

$$M_5 = \left\lfloor \frac{a_3}{a_4} \right\rfloor$$ (4.20)

By examining the Equation (4.16) the number of packets sent by $A$ is large. Examining the Equation (4.18) the size of the message sent by $B$ is also large. These are the limitations in the MSP1 protocol.
4.5. RELIABLE SECURITY PROTOCOLS FOR SENSORS

4.5.2 Multiple Server Protocol 2

The MSP1 protocol was investigated and extended further to reduce the large number of messages sent through the WSN. Thereby the MSP2 was developed as shown in Protocol 4.13. The assumption is made that the servers can communicate through a different network, other than the low bandwidth WSN used by the sensor nodes. For instance, the GNOME platform [179] also has an Ethernet connection it can use as a high speed backbone network to communicate with other GNOME machines. In the MSP2 protocol, the sensor node A only sends one message to a server, denoted as $S_1$. Server $S_1$ then gathers all the required information from the other servers through the server network (rather than the sensor network). Server $S_1$ concatenates the information and sends it to sensor node B. The list of servers may either be a static list, known by the sensor nodes, or it may be a list based on the trustworthiness of the servers.

<table>
<thead>
<tr>
<th>Protocol 4.13 MSP2 — Multiple Server Protocol 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M1</strong></td>
</tr>
<tr>
<td><strong>M2</strong></td>
</tr>
<tr>
<td><strong>M3</strong></td>
</tr>
<tr>
<td><strong>M4</strong></td>
</tr>
<tr>
<td><strong>M4'</strong></td>
</tr>
<tr>
<td><strong>M5</strong></td>
</tr>
<tr>
<td><strong>M6</strong></td>
</tr>
<tr>
<td><strong>M7</strong></td>
</tr>
</tbody>
</table>

4.5.2.1 Security Analysis

If $S_1$ becomes malicious, there are a number attacks that the server can try. The simplest attack is a denial of service, where the server will not gather any extra information from other servers, or does not respond to the sensor node B. If sensor node A does not receive a response from B in a required amount of time, then it should try $S_2$.

Another possible attack $S_1$ can try is to forge the MACs to create legitimate messages from the other servers. As discussed earlier, it is unreasonable to assume a server can forge a message, let alone 2t messages.

If there are $t$ malicious servers, then $S_1$ can contact those servers and not involve the trusted servers. However, the MSP1 is also vulnerable to $t$ malicious servers.

If contacting only one server is still a concern, then a higher level reputation based framework [64] on top of existing authentication protocols to make sure the nodes collaborate with only trustworthy base stations. Another solution is that A can send the first message to $p$ servers, where $p < n$, and each server is allocated servers from which to obtain information (however, this will require more code and logic within the sensor applications).
4.5.2.2 Cost/Complexity analysis

The MSP2 decreases the number of packets sent by $A$. The following equations are used to define how many bytes are sent for each message, $M_1 = 2a_0 + a_1$, $M_2 = 2a_0$, $M_3 = 2a_2 + 2a_3$, $M_4 = a_0 + a_1 + na_2 + na_3$, $M_4' = na_2 + na_3$, $M_5 = a_1 + na_3$, $M_6 = (n+1)a_3$, and $M_7 = a_3$.

The following equations are used to calculate the number of packets sent out.

The size of each message for $M_1$ has not changed between the MSP1 protocol and the MSP2 protocol, however, the number of packets sent has decreased from $O(n)$ to $O(1)$, as shown in Equation (4.21).

$$M_1 = \left\lfloor \frac{2a_0 + a_1}{a_4} \right\rfloor \quad (4.21)$$

In Equation (4.22) a different packet size is used from the messages sent and received by the sensor nodes. This research does not make the assumption that the base stations can only use the sensor network to communicate. If the base stations do use the same network, $a_5 = a_4$.

$$M_2 = (n-1) \left\lfloor \frac{2a_0}{a_5} \right\rfloor \quad (4.22)$$

Equation (4.23) once again assumes the base stations may communicate over a different network other than the sensor network.

$$M_3 = (n-1) \left\lfloor \frac{2a_2 + 2a_3}{a_5} \right\rfloor \quad (4.23)$$

Equation (4.24) and Equation (4.25) has $S_1$ return all information from the KDCs back to the sensor nodes.

$$M_4 = \left\lfloor \frac{a_0 + a_1 + na_2 + na_3}{a_4} \right\rfloor \quad (4.24)$$

$$M_4' = \left\lfloor \frac{na_2 + na_3}{a_4} \right\rfloor \quad (4.25)$$

Equation (4.26) contains the key information, and the cross-checksums.

$$M_5 = \left\lfloor \frac{a_1 + na_3}{a_4} \right\rfloor \quad (4.26)$$

The last two messages are the same as those defined in the proposed MSP1 protocol, so therefore the equations specifying the number of packets is also the same.

Although the MSP2 decreased the number of messages sent by $A$, it is not as reliable
as MSP1. If $S_1$ is down, either $A$ will need to detect this and try $S_2$, or $S_1$ itself will need to be a clustered system. The drawback of a true replicated clustered system is that it is more likely that the system can be compromised, since there are more machines available for an adversary to attack. Another problem if $S_1$ becomes malicious is that it may not return any information from the other servers. Once again, $A$ can try other servers.

### 4.5.3 Analysis of the Multiple Server Protocol 3

The third multiple server protocol is the MSP3. The third protocol was designed for the case where if the key $K_S$ becomes compromised, and the long-term keys $K_{AS}$ and $K_{BS}$ have not changed. The key $K_S$ is important because it can be used in the future to create or renew a session key between $A$ and $B$. The multiple server scenarios have strengthened the security between the nodes and the KDC. Compromised keys between a node and one (or more) servers, does not affect the security of the protocol. An adversary can replay previous (portion of) messages to force the sensor nodes to use the same $K_S$ as before.

The MSP3 protocol was designed to remove this problem. The advantage of the MSP3 over MSP2 is that if $K_S$ is ever compromised and the long term keys $K_{AS}$ and $K_{BS}$ have not changed, then the protocol can be run again safely. The calculation for $\Theta$ was also done so that a replay attack is not possible.

$$\Theta_{Ai} = [A, B, N_A, N_B]_{K_{Ai}}$$
$$\Theta_{Bi} = [A, B, N_A, N_B]_{K_{Bi}}$$

### Protocol 4.14 MSP3 — Multiple Server Protocol 3

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Message</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>$A \rightarrow B$</td>
<td>$A, N_A$</td>
</tr>
<tr>
<td>M2</td>
<td>$B \rightarrow S_1$</td>
<td>$A, B, N_A, N_B$</td>
</tr>
<tr>
<td>M3</td>
<td>$S_1 \rightarrow S_i$</td>
<td>$A, B, N_A, N_B$</td>
</tr>
<tr>
<td>M4</td>
<td>$S_i \rightarrow S_1$</td>
<td>$\Theta_{Bi}, \Phi_{Bi} \oplus K_i, \Theta_{Ai}, \Phi_{Ai} \oplus K_i$</td>
</tr>
<tr>
<td>M5</td>
<td>$S_1 \rightarrow B$</td>
<td>$\Theta_{B1}, \Phi_{B1} \oplus K_1, \ldots, \Theta_{Bi}, \Phi_{Bi} \oplus K_i$</td>
</tr>
<tr>
<td>M5’</td>
<td>$S_1 \rightarrow A$</td>
<td>$\Theta_{A1}, \Phi_{A1} \oplus K_1, \ldots, \Theta_{Ai}, \Phi_{Ai} \oplus K_i$</td>
</tr>
<tr>
<td>M6</td>
<td>$B \rightarrow A$</td>
<td>$cc_B(1), \ldots, cc_B(n), N_B$</td>
</tr>
<tr>
<td>M7</td>
<td>$A \rightarrow B$</td>
<td>$cc_A(1), \ldots, cc_A(n), [N_B]<em>{K</em>{AB}}$</td>
</tr>
<tr>
<td>M8</td>
<td>$B \rightarrow A$</td>
<td>$[N_A]<em>{K</em>{BA}}$</td>
</tr>
</tbody>
</table>

The MSP3 has one more message than MSP2, however, MSP2 has a larger message. This is because of the need to send $A$ and $N_A$ in $M4$, whereas in MSP3 the message $M5$ does not need to send this extra information. The maximum size message in MSP2 can be decreased if another message $M1'$ is sent as shown in Equation (4.28).

$$A \rightarrow B : A, N_A$$

(4.28)
4.6 General Solution

This section will show that the methodology used to make Boyd four-pass protocol into a multiple server protocol, can also be used on other single server protocols. As an example, SPINS protocol (Protocol 4.9) was chosen as the starting point.

The same procedure as applied to the Boyd four-pass protocol is applied to the SPINS protocol, to convert it to a multiple server protocol. The new Multiple Server Protocol for SPINS Protocol 4.15 is proposed. From this example, this section has shown that a different single server protocol can be converted to a multiple server protocol using the same methodology as described in Section 4.5. It was not shown that this is true for all single server protocols, but it is interesting that the methodology could be easily used on another single server protocol.

### Protocol 4.15 Multiple Server Protocol for SPINS

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>$A \rightarrow B : A, N_A$</td>
</tr>
<tr>
<td>M2</td>
<td>$B \rightarrow S_1 : A, B, N_B, [A, B, N_A, N_B]<em>{K</em>{B_1}}, \ldots, [A, B, N_A, N_B]<em>{K</em>{B_n}}$</td>
</tr>
<tr>
<td>M3</td>
<td>$S_1 \rightarrow S_1 : A, B, N_B, [A, B, N_A, N_B]<em>{K</em>{B_1}}, \ldots, [A, B, N_A, N_B]<em>{K</em>{B_n}}$</td>
</tr>
<tr>
<td>M4</td>
<td>$S_1 \rightarrow S_1 : \left[ [K_1]<em>{K</em>{B_1}}, [A, B, N_A, N_B, [K_1]<em>{K</em>{B_1}}]<em>{K</em>{B_1}}, \ldots, [K_n]<em>{K</em>{B_n}}, [A, B, N_A, N_B, [K_n]<em>{K</em>{B_n}}]<em>{K</em>{B_n}} \right]$</td>
</tr>
<tr>
<td>M5</td>
<td>$S_1 \rightarrow B : \left[ [K_1]<em>{K</em>{B_1}}, [A, B, N_A, N_B, [K_1]<em>{K</em>{B_1}}]<em>{K</em>{B_1}}, \ldots, [K_n]<em>{K</em>{B_n}}, [A, B, N_A, N_B, [K_n]<em>{K</em>{B_n}}]<em>{K</em>{B_n}} \right]$</td>
</tr>
<tr>
<td>M5'</td>
<td>$S_1 \rightarrow A : \left[ [K_1]<em>{K</em>{B_1}}, [A, B, N_A, N_B, [K_1]<em>{K</em>{B_1}}]<em>{K</em>{B_1}}, \ldots, [K_n]<em>{K</em>{B_n}}, [A, B, N_A, N_B, [K_n]<em>{K</em>{B_n}}]<em>{K</em>{B_n}} \right]$</td>
</tr>
<tr>
<td>M6</td>
<td>$B \rightarrow A : cc_{B}(1), \ldots, cc_{B}(n), N_B$</td>
</tr>
<tr>
<td>M7</td>
<td>$A \rightarrow B : cc_{A}(1), \ldots, cc_{A}(n)$</td>
</tr>
</tbody>
</table>

4.7 Performance Analysis

When looking at authentication algorithms, a number of different aspects need to be taken into consideration. Apart from the security properties, such as key establishment, key freshness, and key confirmation, a number of performance aspects should also be looked at. In the past, symmetric key algorithm performance was categorised by the number of messages and the number of rounds. However, in sensor networks these are not true indicators of the performance of an algorithm. Other measures such as computational costs, number of bytes and packets sent and received, the amount of memory consumed are also important in a sensor environment.

The performance measurements should take the following attributes into account:

- **The number of bytes encrypted and decrypted by each sensor node, and base station.**
- **The number of hashes calculated, and the number of bytes used in the hash function, and whether the checks are done by the sensor nodes or the base station.**
4.7. PERFORMANCE ANALYSIS

- The number of bytes and packets sent and received, by each sensor node, and by the base station.
- The distances between sensor nodes and the base station, and the number of nodes the messages have routed to get to the destination.
- The amount of resources, such as memory, required to run the algorithm.

Assume the size of the key to be 64 bits and the size of the node id to be 16 bits (assuming no more than 64K sensor nodes in a network), and the size of the nonce to be 8 bits.

The computational costs to the scheme arise because of the encryption, decryption and integrity checking of the keys generated by the servers. There is also the generation of the $K_{AB}$ by both $A$ and $B$. An inefficient aspect of the Boyd four-pass protocol is the need to decrypt the message before the integrity check is done. The protocols have a similar restriction if bits were changed in the $Φ$ value, then it will not be known until both $Φ$ and $Θ$ were calculated. Bogus messages injected by a third party cannot be detected using any computationally efficient methods. The Bellare–Rogaway uses the encrypted messages when performing the integrity check, rather than the decrypted message. A very simple modification to the Boyd four-pass protocol removes this minor limitation, as shown in the Protocol 4.16.

\begin{protocol}
\textbf{Protocol 4.16 Modified Boyd protocol integrity change}
\begin{align*}
M1 & \quad A \rightarrow S : \quad A,B,N_A \\
M2 & \quad S \rightarrow B : \quad [[K_S]]_{K_{AS}},[[K_S]]_{K_{BS}},[[K_S]]_{K_{BS}},[[K_S]]_{K_{BS}},[A,B,[[K_S]]_{K_{BS}},[A,B,[[K_S]]_{K_{BS}},\ldots]
M3 & \quad B \rightarrow A : \quad [[K_S]]_{K_{BS}},[[K_S]]_{K_{BS}},[[K_S]]_{K_{BS}},[[K_S]]_{K_{BS}},N_A,K_{AB},N_B \\
M4 & \quad A \rightarrow B : \quad [N_B]_{K_{AB}}
\end{align*}
\end{protocol}

The size of the messages does not change, and there is the added benefit of the integrity check performed before the decryption of the message. This technique was also extended to be used in the proposed protocols (however, encryption algorithms are then required to be implemented on the sensor nodes). However, it has been shown that the communication costs are almost an order of magnitude more than the computational costs in security systems [127].

The communication costs will be heavily dependent on the topology of the network. Also, different protocols have different communication overheads for the sensor nodes, and the servers. The communication overheads of the different protocol are examined in detail in Section 4.8.
4.8 Comparison

This section will contain a comparison of the proposed protocols with the Chen et al. protocol and the Gong protocol. The computational complexity of the existing multiple server protocols is greater since the key is constructed using a key-sharing mechanism. This has two major drawbacks in a sensor network environment. The first is the amount of extra code (and therefore memory overhead) needed when creating the new key. The second is the additional computation (and therefore extra energy) required when creating the new key. If a threshold scheme such as Shamir’s scheme [139] is used to recover the key, the computational complexity will be $O(t \log_2 t)$. Whereas the proposed protocols use a simple exclusive-or function (as describe in Equation (4.14)) to recover the key, which has a computational complexity of only $O(n)$. Not only does this save on computational cost, but also has the added benefit of requiring less code than a full-blown key-sharing threshold scheme. The advantage of using a full-blown key-share threshold scheme is that $t$ servers can calculate the new session key between $A$ and $B$ if they are the only servers involved in the protocol. However, for performance reasons the same restriction was not placed on the proposed multiple server protocols.

A comparison on the number of cryptographic operations between the existing multiple server protocols and the proposed protocols Figure 4.7 compares the number of cryptographic operations performed by the MSP1 protocol with the two existing multiple server protocols. The Gong multiple server protocol is the most expensive, followed by the Chen multiple server protocol. Both protocols have an $O(n^2)$ number of operations. The proposed protocol has an $O(n)$ number of operations.

![Figure 4.7: Number of Operations for Different Protocols](image-url)
Another comparison is the communication cost of the proposed protocols compared with the Chen et al. and Gong protocols. For simplicity, an assumption is made that the output of the one-way function used to calculate the cross-checksums in the existing multiple server protocols is the same size as the integrity function used in the proposed protocols. Although, as described earlier, the one-way function in the existing multiple server protocols will need to be larger for the system to be secure.

This section will compare the total number of bytes sent by sensor node $A$ for each of the existing multiple server protocols, and MSP1 and MSP2. However, similar calculations as the ones shown here can be used to calculate the impact on the base stations and sensor $B$.

The total number of bytes sent by sensor $A$ in the Gong multi-server protocol is $(n^2a_3 + 2na_0 + na_2 + 2a_0 + 2a_1 + a_3)$, which has a complexity of $O(n^2)$. The number of bytes sent by sensor $A$ in the Chen et al. multi-server protocol is $(n^2a_3 + na_3 + 3a_0 + 3a_1 + a_3)$. The above two cases have a complexity of $O(n^2)$. However, the number of bytes sent by sensor $A$ in MSP1 and MSP2 are $(2na_0 + na_1 + (n + 1)a_3)$ and $(2a_0 + a_1 + (n + 1)a_3)$, respectively. Both of these messages have a complexity of $O(n)$.

Figure 4.8 compares the total number of bits sent out in the entire network, relative to the number of servers used. The Gong protocol is the most expensive, followed by the Chen protocol. Also, the protocols consider for this graph send all their messages over the sensor network, rather than sending some of the messages over a faster backbone network.

![Figure 4.8: Total Number of Bits Sent by Different Protocols](image-url)
CHAPTER 4. RELIABLE SECURITY PROTOCOLS

The existing multiple server protocols have message sizes of size $O(n^2)$, whereas protocols are $O(n)$. It should be noted that the existing multiple server protocols do have added functionality, where the trusted servers are able to calculate the key $K_{AB}$. However, for performance reasons the same restriction was not placed on multiple server protocols.

Table 4.6 describes the costs of the MSP1, where $n$ is the number of servers. The table shows that in the protocol sensor node $A$ transmits the most number of messages. However, even though sensor node $B$ only transmits two messages, the size of one of its messages is based on the number of servers. Hence, the number of bits sent by sensor node $B$ is similar to the number of bits sent by sensor node $A$. The messages and size of messages transmitted by a server $S_i$ is constant and independent on the number of servers in the protocol. The number of messages received and the number of bits received is very similar when comparing sensor node $A$ and sensor node $B$. The message and number of bits received by a server $S_i$ is insignificant compared to all the other messages.

Table 4.6: Performance Measurements for MSP1

<table>
<thead>
<tr>
<th>Node</th>
<th>A</th>
<th>B</th>
<th>$S_1$</th>
<th>$S_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Msgs Tx</td>
<td>$n + 1$</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Msgs Rx</td>
<td>$n + 2$</td>
<td>$n + 1$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Bits Tx</td>
<td>$72n + 32$</td>
<td>$32n + 40$</td>
<td>248</td>
<td>248</td>
</tr>
<tr>
<td>Bits Rx</td>
<td>$144n + 40$</td>
<td>$168n + 32$</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 4.7 describes the costs of the MSP2, where $n$ is the number of servers. The table shows that in the protocol the server $S_1$ transmits the most number of messages. However, even though sensor nodes $A$ and $B$ only transmit several messages, some of their message sizes are based on the number of servers. Hence, the number of bits sent by sensor nodes $A$ and $B$ is still linear. The messages and size of messages transmitted by a server $S_i$ (except $S_1$) is constant and independent on the number of servers in the protocol. The number of messages received and the number of bits received is very similar when comparing sensor node $A$ and sensor node $B$. The message and number of bits received by a server $S_i$ (except $S_1$) is insignificant compared to all the other messages. The number of bits received by $S_1$ is linear in nature.

Table 4.7: Performance Measurements for MSP2

<table>
<thead>
<tr>
<th>Node</th>
<th>A</th>
<th>B</th>
<th>$S_1$</th>
<th>$S_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Msgs Tx</td>
<td>2</td>
<td>2</td>
<td>$n + 1$</td>
<td>1</td>
</tr>
<tr>
<td>Msgs Rx</td>
<td>3</td>
<td>2</td>
<td>$n$</td>
<td>1</td>
</tr>
<tr>
<td>Bits Tx</td>
<td>$32n + 72$</td>
<td>$32n + 40$</td>
<td>$232n - 40$</td>
<td>192</td>
</tr>
<tr>
<td>Bits Rx</td>
<td>$128n + 40$</td>
<td>$128n + 56$</td>
<td>$192n - 152$</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 4.8 describes the costs of the MSP3, where $n$ is the number of servers. The
4.8. COMPARISON

Protocol has many of the same features as MSP2. For instance, the server $S_1$ transmits the most number of messages. The sensor nodes $A$ and $B$ only transmit several messages, some of their message sizes are based on the number of servers. Hence, the number of bits sent by sensor nodes $A$ and $B$ is still linear. However, the message sizes are slightly larger when compared against the previous protocol because of the extra security. The messages and size of messages transmitted by a server $S_i$ (except $S_1$) is constant and independent on the number of servers in the protocol. The number of messages received and the number of bits received is very similar when comparing sensor node $A$ and sensor node $B$. As in the previous protocol, the message and number of bits received by a server $S_1$ (except $S_1$) is insignificant, however, it is slightly larger when compared to the previous protocol. As in the previous protocol, the number of bits received by $S_1$ is linear in nature.

Table 4.8: Performance Measurements for MSP3

<table>
<thead>
<tr>
<th>Node</th>
<th>$A$</th>
<th>$B$</th>
<th>$S_1$</th>
<th>$S_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Msgs Tx</td>
<td>2</td>
<td>3</td>
<td>$n + 1$</td>
<td>1</td>
</tr>
<tr>
<td>Msgs Rx</td>
<td>3</td>
<td>3</td>
<td>$n$</td>
<td>1</td>
</tr>
<tr>
<td>Bits Tx</td>
<td>$32n + 72$</td>
<td>$32n + 88$</td>
<td>$240n - 48$</td>
<td>$192$</td>
</tr>
<tr>
<td>Bits Rx</td>
<td>$128n + 40$</td>
<td>$128n + 72$</td>
<td>$192n - 144$</td>
<td>$48$</td>
</tr>
</tbody>
</table>

The MSP1 puts a larger emphasis on communication costs on sensor node $A$. The other two protocols put greater emphasis on the server nodes. The extra message in MSP3 does not affect sensor node $A$, and for a large number of servers it is insignificant.

The number of cryptographic operations performed by each node in the three protocols shown in the table is the same for each of the protocols. The number of $\Theta$ calculations is $(2n + 4)$ for both $A$ and $B$. The number of $\Phi$ calculations is $2n$ for both $A$ and $B$. For $S_i$ the number of $\Theta$ and $\Phi$ calculations is two. The only nodes that need to create good random numbers are each of the servers $S_i$ and they only create one for each run through of the protocol. Both $N_A$ and $N_B$ can be a simple counter.

Another comparison is the number of packets sent by each of the proposed multiple server protocols, as shown in Figure 4.9. The comparison is done over two different network topologies. The servers either communicate over the sensor network or they have a separate network to communicate over. The MSP1 has the same cost over both network topologies, since every message interacts with a sensor node, and the servers do not need to communicate with one another. Because of the larger number of messages sent by the MSP2 and MSP3, it is natural for the number of packets to be more than MSP1, when all the messages have to be sent on the same sensor network. The increase in the number of servers does not correspond in a noticeable difference between MSP1 and the other protocols (when on the same network). The other two protocols have the advantage of concatenating messages, which the original multiple server protocol does not have. If the servers can communicate over a different network, the number of packets sent by MSP2
and MSP3 stay very close to one another, and sometimes they are the same. This is because of the first protocol having larger message sizes, and the messages need to be segmented sooner than they do in the other protocol. The number of packets sent on the energy-constrained sensor network is significantly larger if using MSP1, when the servers are on different networks.

![Figure 4.9: Number of Packets Sent by Different Protocols](image)

TinyOS has a seven byte overhead when sending a single packet. When comparing the bits sent by each of the protocols, the packet overhead is included. The case of two network topologies is once again covered, as shown in Figure 4.10. The MSP1 has the same cost over both networks. When comparing the protocols, if the servers can communicate over a different network, there is virtually no difference between MSP2 and MSP3. However, the extra overhead of sending data to the server $S_1$, which then sends the same data down to the sensor nodes, causes both protocols to have significantly more overhead when run over the same network.

Table 4.9 compares the existing multiple server protocols with some of protocols. In the table the protocol reliability is investigated, and compare the protocols when the servers use the WSN or a different network. An evaluation is performed of the robustness of the protocols based on security attacks in any environment.

After the detailed analysis of the protocols and the comparison with existing protocols, a conclusion that can be derived that MSP1 should be used in situations where the environment is concerned with reliability. If reliability is not a major concern and if the servers can communicate over a different network, then MSP2 should be used. If the environment is
4.8. COMPARISON

![Figure 4.10: Number of Bytes Sent by Different Protocols](image)

**Table 4.9: Comparison of Multiple Server Protocols**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Gong</td>
<td>Excellent</td>
<td>Bad</td>
<td>Bad</td>
<td>Excellent</td>
</tr>
<tr>
<td>Chen et al.</td>
<td>Excellent</td>
<td>Bad</td>
<td>Bad</td>
<td>Excellent</td>
</tr>
<tr>
<td>MSP1</td>
<td>Excellent</td>
<td>Good</td>
<td>Average</td>
<td>Good</td>
</tr>
<tr>
<td>MSP2</td>
<td>Average</td>
<td>Average</td>
<td>Excellent</td>
<td>Good</td>
</tr>
<tr>
<td>MSP3</td>
<td>Average</td>
<td>Average</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
</tbody>
</table>
CHAPTER 4. RELIABLE SECURITY PROTOCOLS

not multi-tiered, then the most reliable protocol (MSP1) is also the most efficient. If there is concern about the security of $K_S$, then MSP3 should be used. In a complex environment the need for reliability and security puts extra costs on the protocols.

4.9 Implementation

In the previous section a comparison was performed of the multi-server protocols using simulators to measure the number of packets and number of bits sent. However, the simulation was not able to compare both the packets and the bits sent at the same time. One comparison only examined the number of packets. The comparison assumed that each packet was of equal cost, a packet with one byte of data is the same as a packet with 28 bytes of data. The assumption was not meant to be accurate, but rather a close approximation. Another comparison was the total number of bits sent. Once again, this was not meant to be accurate, since two packets sending one byte each is not equivalent to one packet sending two bytes each.

Instead of using simulators, the experiment required an implementation and comparison of the different security protocols on a Crossbow mica2 MPR2600 mote [74]. The protocols were developed using the implementation created for the simulation. The experiment had one sensor attached to a workstation while the other sensors were placed stand-alone. After the running of the protocols, the elapsed time was then sent via the serial connection, to a PC running a Linux® distribution where the experiment has a Java® application reading the TinyOS packet from the serial port, and report that data to the user.

Sensors normally use small keys, encrypting large amounts of data is time and energy consuming. The small keys, such as the key $K_{AB}$ used in the protocols, should be refreshed frequently. The refresh should happen before the average time it will take an adversary to calculate the key using brute force. However, larger keys can exist in the sensor network. The large keys, such as the key $K_S$ can be used to generate the small keys. If the key $K_S$ was small, such as 64 bits, then an adversary is able to use brute force to calculate $K_S$ in a short time, which defeats the purpose of having a long term key.

The experiments originally implemented the protocols using 64 bit keys. However, the mobile health care system, defined in this chapter, requires keys that may exist for weeks or months in the system. So the tests ran the same experiments using a 128 bit keys.

Figure 4.11 shows the total time taken for a single protocol run. Each of the protocols does not use the benefits that a multi-network infrastructure can provide. Every message is either sent to a sensor or is received by a sensor. For completeness, the times for the protocols were also included when 64 bit keys were used. The Gong 128 bit key protocol run was not included, since it took over four seconds to complete the eight server scenario.
There is a notable difference between using an 128 bit key over a 64 bit key. Hence, this protocol should be run rarely in a sensor environment. Also, the proposed protocols generated two keys, whereas both the Gong and Chen et al. protocols only generated one key. In the proposed implementation, the cost of generating the session key was not included. The initial figures from implementing the Gong and Chen et al. protocols showed that the protocols were heavier-weight than the the proposed protocols.

Figure 4.12 compares the times of the proposed protocols if there is only a single network and no multi-tiered network infrastructure. The MSP2 and MSP3 protocols have higher performance costs than the MSP1 protocol. The MSP2 protocol, which is the most secure, has the highest performance cost.

Figure 4.13 compares the times of the proposed protocols on multi-tiered network infrastructure. In the setup the message time results were removed for any message that was using a faster network. A comparison was completed of the performance of the different protocols only on the slower sensor network. The MSP2 and MSP3 protocols have lower performance costs than the MSP1 protocol. As the number of servers increases so does the benefit of using protocols that utilise faster networks. There still is a considerable benefit of using MSP2 and MSP3 when the number of servers is small, such as two or three servers. The MSP3 protocol, which is the most secure, has similar performance costs as the MSP2 protocol. For the extra security and little performance costs, the MSP3 protocol seems to be the best choice.
CHAPTER 4. RELIABLE SECURITY PROTOCOLS

Figure 4.12: Total Time Taken on Single Network

Figure 4.13: Total Time Taken using Different Networks
4.9. IMPLEMENTATION

4.9.1 Performance costs of cryptography

This section will show performance costs of the cryptography. The performance cost of cryptography is separated since they were found to be insignificant when compared to the communication cost. Where the communication can be measured in the milliseconds the computation costs had to be measured in the microseconds range.

Before comparing the different cryptographic primitives, and the benefits that one implementation has over another, the skeleton code that was creates was based on TinyOS 2.x. The skeleton code initialises the sensor node, and after the sensor is initialised the start time was obtained in milliseconds. The experiment then ran a cryptographic primitive in a loop for 2000 iterations, before obtaining a new time. The experiment then subtracted the new time from the initial time to obtain the elapsed time in milliseconds to run the cryptographic primitive for 2000 attempts.

The key establishment protocols uses exclusive-or (xor) to encrypt the new session key. A comparison was performed of this method with other methods of encrypting the new session key for body sensor networks. The experiment had implementations of RC5, SKIPJACK, MD5-HMAC and ECC cryptographic primitives on the mica2 MPR2600 motes using TinyOS 2.x.

Table 4.10 supplies the ratio of time taken for each of the algorithms compared to exclusive-or algorithm. When the algorithm ran on the mica2 mote, over the 2000 iterations it took approximately one millisecond. In the ATEMU simulator it took approximately 7000 instructions.

Table 4.10: Ratio Compared to Exclusive-Or

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mica2</th>
<th>ATEMU</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC5</td>
<td>453</td>
<td>456</td>
</tr>
<tr>
<td>SKIPJACK</td>
<td>739</td>
<td>741</td>
</tr>
<tr>
<td>HMAC–MD5</td>
<td>18400</td>
<td>18500</td>
</tr>
<tr>
<td>ECC</td>
<td>4820000</td>
<td>4920000</td>
</tr>
</tbody>
</table>

There was little difference found between the simulation results and the amount of time an operation takes when put on the mica2 mote. The most notable difference in results was for the ECC algorithm where there was a two percent greater difference, and was not as accurate.

4.9.2 Memory

The memory requirements of the application were also examined, as shown in Table 4.11. The combination of .bss and .data segments use SRAM, and the combination of .text and .data segments use ROM. The .text contains the machine instructions for the application.
CHAPTER 4. RELIABLE SECURITY PROTOCOLS

The .bss contains uninitialised global or static variables, and the .data section contains the initialised static variables.

Table 4.11: Memory Overhead in Bytes on MICA2 Platform

<table>
<thead>
<tr>
<th>Memory</th>
<th>RC5</th>
<th>ECC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
<td>6942</td>
<td>15792</td>
</tr>
<tr>
<td>RAM</td>
<td>367</td>
<td>1283</td>
</tr>
<tr>
<td>.data</td>
<td>14</td>
<td>10</td>
</tr>
<tr>
<td>.bss</td>
<td>353</td>
<td>1273</td>
</tr>
<tr>
<td>.text</td>
<td>6928</td>
<td>15782</td>
</tr>
</tbody>
</table>

4.9.3 Size

The size of the application both with number of lines of code and the size in bytes is important when choosing an algorithm. Table 4.12 list the extra number of lines of code and the extra size in bytes that are added to an application that previously had no security.

Table 4.12: Time Measurements for Different Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Lines of Code</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC5</td>
<td>426</td>
<td>828</td>
</tr>
<tr>
<td>SKIPJACK</td>
<td>617</td>
<td>1798</td>
</tr>
<tr>
<td>HMAC–MD5</td>
<td>427</td>
<td>12714</td>
</tr>
<tr>
<td>ECC</td>
<td>5038</td>
<td>9988</td>
</tr>
</tbody>
</table>

The Lines of Code indicates the complexity for the coder to implement the algorithm. The Size (bytes) indicates the size in bytes of the application.

The RC5 application took considerably more effort than the exclusive-or (xor) application. There was an RC5 implementation for TinyOS 1.x in the TinySEC library; however, it has yet to be ported to TinyOS 2.x. Most of effort with the experimental setup was spent porting the code to the new platform.

The SKIPJACK application had similar problems as the RC5 application. Where there was an implementation for TinyOS 1.x in the TinySEC library but there was not one for TinyOS 2.x. Once again, most of the effort was spent porting the code to the platform.

For the HMAC–MD5 application there was no previous implementations of HMAC–MD5 in any version of TinyOS. In this case the experiments required the code to be obtained from RFC1321 and RFC2104 and then have the code ported to first the nesC language and then to the TinyOS application. This was considerably more effort than either RC5 or SKIPJACK implementations.
4.10. CONCLUSIONS

The ECC application also had similar problems to the RC5 and SKIPJACK implementations. Before the experiment was started, a port of an ECC library, developed for TinyOS 1.x, was done for TinyOS 2.x. The ECC application used a 160 bit points.

The HMAC–MD5 application is the largest; however, the application was a straight port from the RFCs, where the code was not intended for sensors. The ECC application is much larger than a RC5 or SKIPJACK application.

4.10 Conclusions

This chapter and its contributions have mainly tested the Hypothesis 1, which also helps to answer the Research Question 1. A mobile health care system was defined, and the important requirements for the quality of security services in the system were supplied. Key establishment protocols without the assumption of trusting an individual authentication server are needed in sensor environments where clients cannot trust individual servers. If the security services of the system are down, then the system itself may become compromised. This chapter showed that with multiple server protocols, even if one or more servers become unavailable or untrustworthy, it may still be possible for the sensor nodes to establish a good session key. Also, well-known multiple server key establishment protocols were examined and placed into the mobile health care system. The existing multiple server key establishment protocols were found to have large communication overheads and may not be suitable for this environment.

A critical review of single server protocols developed for both traditional and sensor networks was carried out. The performance of single server protocols, including two of the proposed protocols, was analysed in detail. It was also shown that updating keys can be accomplished even if the session key has been compromised. Three multiple server protocols were then created for sensor networks in a complex sensor system containing multi-tiered networks, and provided a detailed analysis and comparison of each the protocols. Implementation of the protocols involved either porting libraries or creating new libraries in TinyOS 2.x. The time elapsed, complexity of the code, and memory requirements are analysed in detail on mica2 sensors. This chapter showed that a symmetric key implementation has advantages over an asymmetric implementation. The proposed protocols have the added benefit of not solely relying on the sensor nodes to generate cryptographically sound pseudo-random numbers, but still using information from each of the sensors to generate the new key. It was shown that the proposed protocols are flexible enough for them to be used with almost any cryptographic primitive and in a range of environments. A map was created of the different scenarios depending on security, performance and availability to the proposed protocols, and showed which protocols work best for each of the different scenarios.

A limitation of the protocols described above is that there is an assumption that long
term keys have been preconfigured into the sensors and authentication servers found in the system. This will add a complication that off the shelf devices cannot be easily added to the system. Configuring security in a complex system may be beyond many users, and if the user is ill, they may not be inclined to learn how to set up or add new devices. The next chapter will describe a new family of protocols that is designed to ease the configuration of the initial keys into the components of the health care system.
5

Establishing Initial Keys

This chapter describes the importance of having a user friendly mechanism for initial key establishment in a mobile health care system. A family of existing key establishment protocols, called password protocols, are analysed. Physiological values from a body are proposed as a viable alternative to passwords, in a password protocol. A new family of key establishment protocols based on password protocols is developed, where keys can be established using changing low entropy data. A security analysis for different physiological data, changing at different rates is discussed. A performance comparison when using RSA and elliptic curves over the new family of protocols is supplied. Another set of protocols are also created where the sensors measure different physiological data can still establish a secure connection. Also, a new group key protocol is proposed where physiological data is used to secure all the nodes. An initial implementation of the new secure protocols is carried out. As in the previous chapter, the implementation of the protocols is tested in a simulator, emulator and real hardware. Also, a detailed analysis of performance results from each of the test-beds is provided.

5.1 Introduction

Wireless sensors and actuators have the potential to significantly change the way people live and interact. As the sensors permeate the environment they can monitor objects, space and the interaction of objects within a space. Sensors can monitor a wide range of diverse phenomena by collecting information such as vibrations, temperature, sound, and light.
Different sensors have different associated costs. For example, a sensor simply detecting light will have different costs to a sensor recording sound. However, less costly sensors can be used to detect a phenomenon before alerting the more expensive sensors to start their monitoring. As the number of heterogeneous sensors increases, so will the amount of interactions between the sensors. Some new key establishment and authentication protocols are proposed that can be successfully used in a Wireless Sensor Network (WSN).

Sensor nodes suffer from limited computational capabilities, battery energy, and available memory. Asymmetric cryptography, in general, is unsuitable for most sensor architectures due to higher computational overhead, as well as energy and memory consumption. Many applications in WSNs do not require encryption, but do require authenticated messages. TinySEC [95] is an example of link layer security, where encryption can be turned off, and Message Authentication Codes (MACs) are added to the packets to ensure the integrity of the messages. Sensor nodes may also contain hardware security. Sensor nodes that only need integrity checking of messages may have hardware that produces MACs, but may not have hardware for encryption.

There are many different types of sensor environments, ranging from large areas covered by sensors, to many sensors in a small area [97]. Different environments have a wide range of different characteristics. For instance, sensors placed in a large open area are not as physically secure as sensors implanted in an individual’s body. The protocols proposed in this chapter are designed for body sensor networks; however, these protocols can also be applied to other environments that have similar security characteristics found when using body sensor networks.

Context awareness is an important aspect of body sensor networks. For instance, blood pressure increases due to exercise is normal. However, the blood pressure increases while at rest could mean a serious medical condition. Sensors on the body may not just measure the physiological values, but also the body motions, and can lead to a situation, where number of different sensors that will need to communicate with each other [165].

Mobile health care systems were initially described in Section 2.1. A specific example and more detailed description of a mobile health care system is supplied in this chapter. The previous chapter described home sensors that can be found in a mobile health care system. This chapter defines how the body sensors, home sensors and the hospital can interact.

Figure 5.1 describes a Body Sensor Network (BSN) communicating with a mobile phone to connect to the Internet, and thus sending data to a hospital. This allows patients to be sent home and to their work, and still enabling the medical staff to monitor them. The mobile phone can be replaced with a PDA, or other devices that can connect the sensors with the Internet [61].

Some of the data recorded from body sensors include the heart rate, blood pressure,
5.1. INTRODUCTION

Figure 5.1: Architecture and Service Platform of a BSN for Telemedicine and m-Health

Temperature, and blood oxygen [11, 184]. Table 5.1 shows the typical range that is sent for each of the physiological values. The node information is the location of the sender node (8 bits), a MAC (the MAC size is the same as the size of the physiological data), and a counter to stop replay attacks (32 bits). Information such as the location of the receiving node is not included in the node information, since it is a part of the packet header as described in TinyOS [102]. The heart rate values require only 2 bits per second; however, the node information increases the data rate to 10 bits per second. For the other physiological values including the node information, there is only a need for 2 to 3 bits per second. This work assumes uses the data rate of 10 bits per second.

Table 5.1: Body Sensor Data Rate Range

<table>
<thead>
<tr>
<th>Signal Type</th>
<th>Depth</th>
<th>Node Info.</th>
<th>Msg Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart rate</td>
<td>8 bits</td>
<td>56 bits</td>
<td>10 msgs/min</td>
</tr>
<tr>
<td>Blood Pressure</td>
<td>16 bits</td>
<td>64 bits</td>
<td>2 msgs/min</td>
</tr>
<tr>
<td>Temperature</td>
<td>16 bits</td>
<td>64 bits</td>
<td>1 msgs/min</td>
</tr>
<tr>
<td>Blood oxygen</td>
<td>16 bits</td>
<td>64 bits</td>
<td>1 msgs/min</td>
</tr>
</tbody>
</table>

As initially described in Section 2.1.5.8, environmental data has been used as the only source of secret information to establish keys between different nodes [14–16, 130, 173]. The major benefit of using environmental data is that body sensors can use the environmental data to authenticate that another sensor is also on the same body, and not
a sensor on another individual. However, the approach has a number of problems, some of the problems are:

- Every node must have the same type of sensor, and must be able to sense the same phenomenon. This may be cost prohibitive or too bulky.
- Only cryptographically strong environmental data can be used.
- BSNs can contain heterogeneous set of sensor nodes measuring different physiological values, therefore existing protocols, which use environmental data, cannot be used to secure communication between the nodes.
- The environmental values can become compromised, thus the new session key is also compromised.

These problems limit the use of environmental data for establishing keys to only a few cases.

This chapter develops several protocols to address these problems. Health sensors are used as an example of a network where these protocols can be used. The proposed protocols extend traits from a two party password-authenticated key exchange protocol [34] and server-based key establishment protocols [21, 30]. The proposed protocols do not require traditional encryption to transport the new session key. The rest of this chapter will show that the sensor nodes can establish keys even if no previous shared keys exist between them.

### 5.2 Notation

The notation used in this chapter is initially described in Table 4.1. However, additional terms are required when defining protocols using environmental data, these terms are described in Table 5.2.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rightarrow m$</td>
<td>Another way to define sending of message $m$.</td>
</tr>
<tr>
<td>$\Upsilon$</td>
<td>Cryptographically strong key generated from the environment.</td>
</tr>
<tr>
<td>$\pi$</td>
<td>Non-changing low entropy key (e.g., a password).</td>
</tr>
<tr>
<td>$\Psi$</td>
<td>Changing low entropy key (e.g., Inter–Pulse Interval of the heart).</td>
</tr>
</tbody>
</table>

### 5.3 Environment Information

As was previously introduced in Section 2.1.5.8, this thesis uses the generic name *Secure Environmental Value* (SEV) to refer to sensed data that can be obtained by sensors from...
their environment. This data is usually hard to obtain through other means. An example of an environment that SEVs may be used in includes the human body, where it is difficult to attach a device on the body without the knowledge of the person.

Sensors found on the body can form a BSN to measure the physiological values found in individuals [10]. Recent research in BSNs has shown that environmental information found in the body can be used to secure communication between sensors nodes [14–16, 130, 173]. Health sensors can use Inter-Pulse Interval (IPI) [130] or Heart Rate Variance (HRV) [16] as good sources for cryptographically random numbers and the physiological values can be used as a one-time pad. Protocols that used these physiological values to encrypt a new key between a sensor pair have been developed [14, 173]. For instance, Venkatasubramanian et al. [173] used a single message to send a new key to the neighbouring sensor node, as shown in Protocol 5.1.

**Protocol 5.1 Venkatasubramanian BSN protocol**

\[ A \rightarrow B : N_A, [N_A]_{K_{AB}}, K_{AB} \oplus \Upsilon \]

The new key \( K_{AB} \) is encrypted with a cryptographically strong physiological value \( \Upsilon \), which is only known to sensors on a particular person. Sensor node \( B \) validates that \( K_{AB} \) is valid (and not generated by an adversary) by verifying the MAC of \( N_A \).

As reported earlier, Venkatasubramanian and Gupta noted that finding additional cryptographically sound physiological values is still an open research problem. Another problem is that all the protocols developed with physiological values require all the sensor nodes to be able to measure the same phenomena. Only cryptographically strong physiological values, such as IPI and HRV, can be used. Also, modern wireless technology (ultra wideband — UWB, radar [162]) may be used to remotely capture the heart rate and could cause security risks when only using IPI and HRV to secure the communication. Other cryptographically weaker physiological values, such as blood pressure, and iron count, are less susceptible to those remote attacks. Key establishment protocols that increase the number of available environmental or physiological values will enhance this research area.

### 5.3.1 Security Goals in BSNs

The use of a PIN code or a password is not applicable to BSNs since many of the sensors do not have a user-interface. Sensors also may be placed in hard to reach places, with some of the sensors implanted into the body. To complicate matters, the sensors may harvest energy directly from the body [94], thus allowing the sensors to exist for long periods of time. Updating keys is therefore an important requirement. An efficient method to update keys is to use environmental data. Environmental data in body sensors can be used for authentication, since a sensor on one body will most likely be measuring physiological data that is different from sensors on another body. However, there are a number of
problems with this approach. If the Venkatasubramanian et al. protocol is used, then only cryptographically strong SEVs can be used. The other problem is that the SEV may become compromised in the future. Further difficulty is that all the sensors on the body need to be measuring the same SEV. The goal of the proposed protocols is to use cryptographically weak SEVs and be secure against compromised SEVs. Additionally another requirement for the protocols is that the sensors may not measure the same SEVs. Section 5.4 contains suitable protocols where the sensors measure the same SEV, whereas Section 5.7 contains proposed protocols when the sensors measure different SEVs.

Many of the existing key establishment protocols are designed for open environments where a sensor node can be easily compromised. There is no secure mechanism available to update the keys. Keys in sensor networks are usually 64 bits in size, and they may become easily compromised. Sensors found in BSNs may be operational for years, especially if the sensors are implanted in the body of an individual. WSN key establishment schemes, such as the random key establishment scheme, are designed for a large sensor network, and are still secure if the number of sensors compromised is below the threshold. These characteristics do not match that of the BSNs. Another drawback is that the shared keys cannot be used for entity authentication, since the same keys can be shared by more than a single pair of nodes.

5.4 Securing Sensors Measuring the Same SEVs

Even though PINs and passwords may not be used in sensors, this section will show that password protocols can be used. Passwords have low randomness, and therefore have similar characteristics to many SEVs. A four digit PIN contains less than 14 bits of randomness and can be used in a password protocol. A typical password length of eight characters has less than 48 bit of randomness, if upper and lower case letters as well as the digits 0 to 9 are randomly chosen. The suitability of password protocols is investigated for the sensor environment. Password protocols have the special property of allowing secrets with small entropy to be used for key establishment. Password protocols are designed so that both off-line and on-line attacks are not feasible. A feature or by-product of most password protocols is that if the original key is compromised, then the new key will not be compromised.

Key sizes in sensor networks are small, normally 64 bits, so that the encryption or integrity tests do not consume too much energy [95, 127]. Small key sizes lead to the need to update keys on a regular basis. Another aspect of a heterogeneous sensor network is that different sensors measure different environmental data. There are also sensors that can measure more than one environmental phenomenon.
5.4. SECURING SENSORS MEASURING THE SAME SEVS

5.4.1 Modified Encrypted Key Exchange Protocol using SEVs

There are situations where the environmental value may be obtained after a period of
time. For instance, encrypting data may be too energy exhaustive, hence the SEV may be
sent out later, in the clear, to a remote site. Another reason may be that the environmental
phenomenon moves from a concealed location to a universally observable location. An
example of a two party protocol where the new key will not be compromised if the old
key becomes compromised is the Encrypted Key Exchange EKE protocol [22], as shown in
Protocol 5.2.

**Protocol 5.2** Diffie–Hellman-based EKE protocol

<table>
<thead>
<tr>
<th>Shared Information</th>
<th>Generator g of G where $p - 1 = qr$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$B$</td>
</tr>
<tr>
<td>$r_A \in_R \mathbb{Z}_p$</td>
<td>$r_B \in_R \mathbb{Z}_p$</td>
</tr>
<tr>
<td>$t_A = g^{r_A}$</td>
<td>$K_{AB} = t_A^{r_B}$</td>
</tr>
<tr>
<td>$n_A \in_R {1, \ldots, 2^L}$</td>
<td>$n_B \in_R {1, \ldots, 2^L}$</td>
</tr>
<tr>
<td>$K_{AB} = t_B^{r_A}$</td>
<td>Verify $n_A$</td>
</tr>
<tr>
<td>$[[n_A,n_B]]<em>{K</em>{AB}}$</td>
<td>Verify $n_B$</td>
</tr>
<tr>
<td>$[[n_A]]<em>{K</em>{AB}}$</td>
<td>$[[n_B]]<em>{K</em>{AB}}$</td>
</tr>
</tbody>
</table>

The EKE protocol contains four messages. Node $A$ sends the first message to node $B$,
the message contains the location of $A$ (the location value is in the clear), and the first
part of Diffie–Hellman, $t_A$, is encrypted by the password $\pi$. After the first message is sent,
node $B$ will calculate the second part of the Diffie–Hellman scheme and hence be able to
calculate the session key $K_{AB}$. Node $B$ then sends the second part of the Diffie–Hellman
scheme encrypted by the password $\pi$ to node $A$. The nonce $n_B$ is also sent, encrypted
by the session key $K_{AB}$. The last two messages authenticate both $A$ and $B$, as well as
confirming that they have the session key $K_{AB}$. The encryption of $t_A$, $t_B$, $n_A$, and $n_B$ can be
implemented with an exclusive-or function [22]. Boyd et al. [33]suggest that the size of $p$
only needs to be 160 bits in traditional networks. RSA is suitable in sensor networks if the
exponents are small.

The EKE protocol is modified to replace the password with a value derived from
physiological data, as shown in Protocol 5.3. The $\Psi_1$ and $\Psi_2$ values in the protocol below
are generated from two different SEVs. This is the major difference between the two
protocols, where in the above protocol the password will not change while the protocol
is running. When using physiological data the low entropy key can change while the
protocol is running.
The modified EKE protocol also contains four messages. Node $A$ sends the first message to node $B$, the message contains the location of $A$, and the first part of Diffie–Hellman, $t_A$, is encrypted by the weak key $\Psi_1$. After the first message is sent, node $B$ will calculate the second part of the Diffie–Hellman scheme and hence be able to calculate the session key $K_{AB}$. Node $B$ then sends the second part of the Diffie–Hellman scheme encrypted by the weak key $\Psi_2$ to node $A$. The nonce $n_B$ is also sent, encrypted by the session key $K_{AB}$. The last two messages authenticate both $A$ and $B$, as well as confirming that they have the session key $K_{AB}$. The encryption of $t_A$, $t_B$, $n_A$, and $n_B$ can be implemented with an exclusive-or function [22].

Depending on which environmental value is measured, and how long the protocol will run, different SEVs may be used for the request and response. However, if the SEV stays constant throughout the running of the protocol, then both $\Psi_1$ and $\Psi_2$ will be the same. The EKE protocol is designed for a constant password throughout the running of the protocol, so similar or same data for both $\Psi_1$ and $\Psi_2$ will not adversely affect the protocol.

The EKE protocol was originally designed to handle small entropy secrets, so that off-line and on-line dictionary attacks are unfeasible for an adversary. Another useful feature is that even if the secrets $\Psi_1$ or $\Psi_2$ are compromised or available freely after the running of the key establishment protocol, the session key $K_{AB}$ will remain secure and safe.

Both nonce $n_A$ and $n_B$ are cryptographically strong random numbers, allowing the exclusive-or function to be used for encryption. If any nonce was not cryptographically strong then either $n_A \oplus K_{AB}$ or $n_B \oplus K_{AB}$ operation would allow an adversary to significantly reduce the number of valid $K_{AB}$ values. A characteristic of the EKE protocol is that the nonces are never sent out in the clear, since the nonces are used to encrypt the new key $K_{AB}$.
5.4. SECURING SENSORS MEASURING THE SAME SEVS

5.4.2 Proposed Protocol using SEVs for Nonces

Communication is an expensive operation in sensor networks. An investigation was performed in previous sections to find ways of reducing the number of messages and the number of bits sent between the nodes. In this section the use of SEVs has been expanded to not only be the small entropy key, but also to replace the nonces. This gives the added advantage of removing a message. However, if the nonces are not cryptographically strong, a stronger encryption mechanism than the exclusive-or is required. A protocol has been proposed called SEVs for Nonces (SEVN) as defined in Protocol 5.4, extends the EKE protocol by incorporating SEVs as nonces. The total number of messages is reduced to three.

<table>
<thead>
<tr>
<th>Protocol 5.4 SEVN — protocol using SEVs for Nonces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared Information: Generator $g$ of $G$ where $p - 1 = qr$</td>
</tr>
<tr>
<td>$A$</td>
</tr>
<tr>
<td>$r_A \in_R \mathbb{Z}_p$</td>
</tr>
<tr>
<td>$t_A = g^{r_A}$</td>
</tr>
<tr>
<td>$K_{AB} = t_A^{r_B}$</td>
</tr>
<tr>
<td>$K_{AB} = t_B^{r_A}$</td>
</tr>
<tr>
<td>Verify $\Psi_3$</td>
</tr>
<tr>
<td>$\Psi_3 = h(SEV, 3)$</td>
</tr>
<tr>
<td>$\Psi_4 = h(SEV, 4)$</td>
</tr>
<tr>
<td>$B$</td>
</tr>
<tr>
<td>$r_B \in_R \mathbb{Z}_p$</td>
</tr>
<tr>
<td>$t_B = g^{r_B}$</td>
</tr>
<tr>
<td>$\Psi_4 = h(SEV, 4)$</td>
</tr>
</tbody>
</table>

In the proposed protocol a technique [127] can also be used to reduce communication cost in security protocols. Perrig assumed that communicating parties, such as $A$ and $B$, know each other’s counter values, which can then be used as nonces. So the nonces do not need to be added to each message. However, if messages get lost, the shared counter state can become inconsistent. Perrig et al. suggested protocols to synchronise the counter between the nodes. The synchronisation protocols add extra overhead to sensors, and in a lossy environment there may be a need for a large number of synchronisation attempts. Instead of a counter or a nonce, this thesis proposes that there may be cases where an SEV can be used. When Perrig et al. suggested using counters, the counters were not considered to be secret. However, in the EKE protocol the encryption algorithm can be an exclusive-or function, which will require the nonce to be a secret random number. If the SEV is not cryptographically strong, then another function other than the exclusive-or function is required. Also, if the SEV is the same throughout the running of the protocol, then the SEV will need to be modified such as a number added to the SEV. The SEV used throughout the protocol may be a hash of the original SEV, so $\Psi_1 = h(SEV, 1)$, $\Psi_2 = h(SEV, 2)$, $\Psi_3 = h(SEV, 3)$, and $\Psi_4 = h(SEV, 4)$. 

117
5.4.3 Attack and Security Weakness

A security weakness exists in both the EKE protocol and the SEVN protocol. The security weakness occurs if the same environmental value is used over multiple runs of the protocol. The same environmental data may be used over multiple protocol runs to conserve energy, or may be the environmental data stays constant over a period of time. When environmental data is used to encrypt $t_A$ and $t_B$, an adversary can use brute force to decrypt $[[t_A]]_{\Psi_1}$ and $[[t_B]]_{\Psi_2}$ over all the possible values of $\Psi_1$ and $\Psi_2$ and check if the candidate results are valid. If they are not then that particular SEV can be removed from the list of valid environmental values. If the same SEV is used multiple times, then the SEV can become compromised, and the adversary can perform a man-in-the-middle attack on the protocol. The encryption mechanism in the Venkatasubramanian BSN protocol (Protocol 5.1), does not suffer from this attack since the data that is encrypted ($\Upsilon$) cannot have any invalid values. However, Venkatasubramanian BSN protocol also requires that the SEV is cryptographically strong.

Encryption of data using a weak key introduces a brute force style attack in this protocol. An adversary guessing the SEV values can attempt to decrypt $[[t_A]]_{\Psi_1}$ and examine whether the resulting plaintext is a valid Diffie–Hellman ephemeral value. For instance, if the decryption with a candidate $\Psi_1$ results in a string whose value is greater than $p$, then that candidate is invalid.

If the $\Psi$ does not change, then extra care will need to be taken. Warning flags should be raised if there are a large number of failures when trying to establish keys. A solution to address the above problem has been developed, which involves adding a number $x$ randomly to the encrypted value, so that the encrypted value can be above $p$. The value for $x$ can be $(2^N - p)$. The receiver will decrypt the message and if it is above $p$, then it will need to subtract $x$. If an adversary tries to decrypt the message, it does not know if the message is invalid or not. If $(t_A + x)$ is below $p$, then the sender only transmits $t_A$ since the receiver does not know if the decrypted value has had the value $x$ added onto it. If $p = (2^N - 1)$, then the value for $x$ is equal to 1. In this case only one $t_A$ will ever be compromised when $t_A = (p - 1)$. However, this technique becomes useful to an adversary if the value for $p$ was chosen to be closer to $(2^{N-1} + 1)$. In this case almost every possible $t_A$ value can be checked for validity. The degree of usefulness of this technique in a sensor environment increases as $p$ becomes closer to $(2^{N-1} + 1)$.

Jaspan [91] created a formula to calculate the strength of the protocol. A slight modification to the formula to measure the percentage of keys that would become compromised is shown in Equation (5.1).

$$\kappa = (1 - (p/2^N)^M) \times 100 \%$$ (5.1)
5.4. SECURING SENSORS MEASURING THE SAME SEVS

Where \( M \) is the number of messages encrypted by the SEV. Jaspan used the example when \( p \) differs from \( 2^N \) by \( 1/10000 \), and suggested that a valid value for \( M \) is 1000 messages. In this case around 10% of the SEVs will be invalid. However, the figures for \( M \) may be reasonable in traditional networks, but do not make sense in the sensor network. For instance, an adversary can wait for 1000 messages to be sent by the sensors. If that is not possible, the adversary can send 1000 requests to the sensors, thus obtaining the 1000 responses. The problems with this attack are:

- Energy scarce sensor nodes do not have enough power to receive that many messages. If an adversary did attempt this attack, the sensor node would become non-functional within that period.
- The adversary would have launched a quite effective denial of service attack since they need to occupy the radio channel for such a long time. Thus detecting this type of attack is simple.
- The SEV will need to stay constant during the period of the attack.
- The adversary must always be able to be within attacking distance from the sensors. Body sensors can move to different rooms, or to different continents, making it difficult to perform such an attack.
- Even after one forgery attempt, the adversary will still need to receive an encrypted plaintext message from the sensor node.

An examination of both types of attacks is described further in detail. Assuming the worst case, the SEV does not change for an extended period of time, there are a large number of messages sent, and the adversary has access to every message. However, in the case of the passive adversary, an encrypted plaintext message sent by the sensor is not needed. As shown in Figure 5.2, there is a dramatic increase in the percentage of invalid SEVs that an adversary can detect if a \( p \) is chosen to be closer to \( 2^N - 1 + 1 \). By choosing \( p \) wisely in the EKE protocol as well as in the proposed SEVN protocol the security can be strengthened significantly, and the time-frame to remove a significant number of SEVs is large enough so that the SEV should have changed in that period.

The SEVN protocol does have a problem if the SEVs do not change over multiple executions of the protocol. If a new key, \( K'_{AB} \), is required between sensor nodes \( A \) and \( B \), then the proposed protocol should be rerun. However, if the SEV is the same across multiple runs and \( K_{AB} \) becomes compromised (even in the future), then an eavesdropping adversary can calculate \( K'_{AB} \). If the adversary knows \( K_{AB} \), and the adversary has captured the protocol run to generate \( K_{AB} \), then the adversary can calculate the value of \( \Psi_3 \) and \( \Psi_4 \) off-line using brute force for all the valid SEVs. The EKE protocol does not suffer from this since \( K_{AB} \) is never used to encrypt the SEV. So if \( K_{AB} \) is compromised, it will not in turn give away any information about the SEV.

There is a large drawback to using exclusive-or as the encryption algorithm in the EKE
protocol. The problem is that to encrypt a 64 bit number using exclusive-or, the encryption key also needs to be 64 bits. This leads to a greater size for the messages. However, since the SEVN protocol, never needs to send out the nonces, it is only the size of the SEV that needs to be encrypted, which will normally be significantly less than the size of the key $K_{AB}$.

The randomness of the environmental data can be much lower than the randomness of the keys used in traditional systems. If $x$ is the size of randomness in bits, then an adversary has $1 \text{ in } 2^x$ chances in blindly forging a valid message. The adversary, if using brute force, should be able to succeed in $2^{x-1}$ tries. A brute force attack would have the adversary trying to flood the channel with forgeries, but on a 19.2 kb/s channel (and 163 bit elliptic curve numbers), they can only perform less than 30 forgery attempts per second (for the optimal scenario). Sending $2^{31}$ packets would take over 2 years. An examination of the main issues with this attack will now be supplied.

Not all sensors have the bandwidth of 19.2 kb/s, which is considered high for many sensor environments. For instance, recently developed sensors can only send at one bit per second [169]. As shown in Table 5.3 this makes a large difference on how long it will take to compromise the SEV. An assumption is made that the computational time to perform Diffie–Hellman, as well as sensing the environmental data, is insignificant. Another assumption is that after each forgery attempt, the attacked sensor will send a packet of encrypted well-known data back to the adversary. The table compares the average time it will take to find the SEV, provided the SEV stays constant throughout
the attack. If the SEV changes every couple of seconds, then the SEV can have a small randomness. However, if the SEV changes once every day, and has a small randomness, then a check for a denial of service attack may be required before the adversary can calculate the SEV. For instance, with 10 bits of randomness, it will take on average 512 attempts before an adversary can discover the SEV. Another solution may be to have the sensor produce a random delay when sending the response, so that the number of requests an adversary can send decreases even further. It should be noted that even if the SEV is discovered, any existing keys will not be compromised.

Table 5.3: Comparison of Different SEV Strengths at 10 b/s

<table>
<thead>
<tr>
<th>Bits of Randomness</th>
<th>Average Time to Break</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2.4 days</td>
</tr>
<tr>
<td>20</td>
<td>6.7 years</td>
</tr>
<tr>
<td>30</td>
<td>7.2 thousand years</td>
</tr>
<tr>
<td>40</td>
<td>7.3 million years</td>
</tr>
<tr>
<td>50</td>
<td>7.8 billion years</td>
</tr>
<tr>
<td>60</td>
<td>8.3 trillion years</td>
</tr>
</tbody>
</table>

As the bandwidth increases, the number of messages that a node can theoretically receive also increases. Table 5.4 compares different SEV strengths and the average time it will take to compute the SEV, when using 19.2 kb/s. An assumption is made that the sensors have enough energy to compute a large number of messages in a short period of time. Another assumption that is made is that there is no check performed by the sensors to detect that a large number of key establishment requests have arrived, and that an encrypted message of well-known text is sent after the key is established.

Table 5.4: Comparison of Different SEV Strengths at 19.2 kb/s

<table>
<thead>
<tr>
<th>Bits of Randomness</th>
<th>Average Time to Break</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10 seconds</td>
</tr>
<tr>
<td>20</td>
<td>1.2 days</td>
</tr>
<tr>
<td>30</td>
<td>3.8 years</td>
</tr>
<tr>
<td>40</td>
<td>4 thousand years</td>
</tr>
<tr>
<td>50</td>
<td>4.3 million years</td>
</tr>
<tr>
<td>60</td>
<td>4.6 billion years</td>
</tr>
</tbody>
</table>

A disadvantage with two party protocols is that every sensor does not measure the same phenomenon. Sensors on the body may not just measure the physiological values, but also the body motions, and can lead to a number of different sensors that will need to communicate with one another [12, 165].

Even though exclusive-or and block cipher symmetric cryptography is suitable when using RSA, it is not suitable when converting to elliptic curves [33, 114]. However, pro-
Protocols have been developed that can encrypt the ephemeral keys in a safe manner. Since these encryption algorithms can be directly mapped to elliptic curve cryptography, the next section will briefly discuss these protocols.

5.5 Analysis of RSA and Elliptic Curves in BSNs

Previous literature [116, 178] suggested that RSA is not suitable for sensors, and elliptic curves should be used. This section will investigate if an RSA implementation is more efficient than an elliptic curve implementation, for the new proposed protocols. Another case that is covered is where there is a sensor application that contains an elliptic curve implementation and there is not enough memory for an RSA implementation. In this case suitable protocols will be described that can be used with elliptic curves.

5.5.1 Analysis of Other RSA based Password Protocols

In traditional networks, password protocols can use different encryption algorithms, thus leading to several variants of the original EKE protocol. The variant protocols include the PPK protocol, PAK–R protocol [114], and the SPEKE protocol [88].

Common sizes used by these protocols are: \( p \) is 1024 bits, \( r \) is 864 bits and \( q \) is 160 bits.

The technique used by the PAK protocol for its RSA based protocol uses:

1. The \( P = H(A, B, \Psi)^r \) for the calculation performed to map \( \Psi \) into the group. The hash of \( A \), \( B \), and \( \Psi \) is taken to the power \( r \).
2. The \( r_A \in_R \mathbb{Z}_q \) for a random value found in the \( \mathbb{Z}_q \) field.
3. The \( t_A = g^{r_A} \) for the ephemeral key created for the Diffie–Hellman operation.
4. The \( m = t_A P \) for the message created, to decrypt the message the other node can divide \( P, m/P \). An adversary will not be able to discover the value for \( t_A \) unless they know \( \Psi \). Also, the adversary will not find any invalid decrypted values, which removes the partition attack problem.

This protocol does suffer from the extra computational costs attributed to the larger exponents, and thus leading to much larger message sizes. In traditional networks, the designs of protocols err on the side of caution. Sensor networks are not afforded this luxury.

Another technique to map the key to the group in the RSA base algorithms is to use \( P = g^\Psi \), and the shared secret is \( P^{r_A r \Psi} \). However, flaws were found in this method [121].

Another technique is to use \( P = \Psi^2 \), where \( \Psi \) is interpreted as an element of \( \mathbb{Z}_p^* \). The shared secret is \( P^{r_A t_A} \). This is used by the SPEKE protocol, as shown in Protocol 5.5.

1. \( P = \Psi^2 \). Map \( \Psi \) into an element of \( \mathbb{Z}_p^* \) of prime order \( q = (p - 1)/2 \).
2. Select a random exponent value \( r_A \). Jablon [88] suggested that the size of \( r_A \) to be 160 bits; however, no security proof could be defined. Mackenzie [113] provided a security proof; however, the exponent needs to be of order \( \mathbb{Z}_q \), so that \( r_A \in \mathbb{Z}_q \). This causes the size of the exponent to be 1024 bits.

3. \( t_A = P^{r_A} \). The ephemeral key is created for the Diffie–Hellman algorithm.

A problem with the above constructs is that the size of the messages will be over 1024 bits. In an energy constrained and low bandwidth environment, this is not suitable. However, when using the Jablon [88] method of generating the exponent, this will limit the computational expense. To limit the message sizes caused by RSA, elliptic curve cryptography has been proposed for sensor networks [116, 178].

Protocol 5.5 SPEKE protocol

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_A \in_R {1, \ldots, 2^L} )</td>
<td>( r_B \in_R {1, \ldots, 2^L} )</td>
</tr>
<tr>
<td>( t_A = P^{r_A} )</td>
<td>( t_B = P^{r_B} )</td>
</tr>
<tr>
<td>( K_{AB} = t_B^{r_A} )</td>
<td>( n_B \in_R {1, \ldots, 2^L} )</td>
</tr>
<tr>
<td>( n_A \in_R {1, \ldots, 2^L} )</td>
<td>( H_1(t_A, t_B, K_{AB}, \Psi) )</td>
</tr>
<tr>
<td>Verify ( \Psi_B )</td>
<td>( H_2(t_A, t_B, K_{AB}, \Psi) )</td>
</tr>
</tbody>
</table>

5.5.2 Comparison of RSA and Elliptic Curves

Many RSA based password protocols can be generalised to use elliptic curves [114]. Using a straightforward conversion of RSA to elliptic curves on the EKE protocol, a modified protocol is created and described in Protocol 5.6.

The modified protocol is similar to the EKE protocol, except that instead of exponents, elliptic curve point multiplications are used. However, encryption of an elliptic curve point using a low randomness is not as simple as encrypting an element of \( \mathbb{Z}_p^* \). The problem is that an adversary guessing \( \Psi_1 \) can attempt to decrypt \( [t_A]_{\Psi_1} \) and examine whether the resulting plaintext is a valid point on the curve. A symmetric key algorithm matched to the elliptic curve group is required.

To convert the above RSA implementation, the \( \Psi_1 \) will need to be mapped to an elliptic curve point. A general procedure for this can be found in IEEE P1363.2: Standard Specifications for Password-Based Public-Key Cryptographic Techniques [83]. A simplified version of
CHAPTER 5. ESTABLISHING INITIAL KEYS

Protocol 5.6 Modified EKE protocol with elliptic curve

Shared Information: Generator $g$ of $G$ where $y^2 = x^3 + ax + b$

$A$

$B$

$r_A \in \mathbb{R} \mathbb{Z}_p$

$t_A = r_A g$

$K_{AB} = r_A t_B$

$t_B = r_B g$

$K_{AB} = r_A t_B$

Verify $n_B$

Verify $n_A$

the procedure is shown below.

1. Set $i = 1$.
2. Compute $w = h(A, B, \Psi_1, i)$.
3. Set $\alpha = w^3 + aw + b$.
4. If $\alpha = 0$ then the point is $(w, 0)$.
5. Find the minimum square root of $\alpha$, and call it $\beta$. Can use the method found in IEEE P1363 [84] (Appendix A.2.5).
6. If no square root exists, set $i = i + 1$, and go to Step 2.
7. The elliptic curve point is $(w, \beta)$.

The above algorithm is non-deterministic for different $\Psi_1$ values. If a square root is not found, then the algorithm will loop back to the second step to compute a new value for $x$. In an environment where a sensor scavenges energy to perform an operation, it is not suitable to have a non-deterministic algorithm.

Mapping a variable into a point on an elliptic curve allows the conversion of many RSA password protocols to an elliptic curve password protocol.

1. Mapping $\Psi$ into the field. When using RSA, the SEV can be naturally mapped into the field. This could either be a direct modulus, or a modulus of the hash of the SEV. However, mapping the SEV onto a point requires more work as shown by the following equation $P = f(A, B, \Psi)r$. The function $f$ is the non-deterministic method to map a number onto a point in an elliptic curve, as described above. The point is then multiplied by $r$, to place it into the correct group.
2. Both the RSA and the ECC implementation require a random value in the field $\mathbb{Z}_q$ to obtain an $r_A$ as shown by $r_A \in \mathbb{R} \mathbb{Z}_q$.
3. The ephemeral value for Diffie–Hellman is calculated, for ECC, that is $t_A = r_A g$. The RSA algorithm involves an exponent.
4. Finally, the message is created. In the RSA case, $P$ is multiplied to $t_A$, whereas in the ECC case the following equation is used $m = t_A + P$. The message $m$ is created, to decrypt the message the receiver can subtract $P$, since $t_A = m - P$. An adversary will not be able to discover the value for $t_A$ unless they know $\Psi$. Also, the adversary will not find any invalid decrypted values, which removes the partition attack problem.

However, when converting this into an elliptic curve-based construct, the similar problems occur as they did for the PAK family of products. The SEV will need to be mapped onto an elliptic curve point, which is non-deterministic.

A technique is developed to make the protocol deterministic, where the elliptic curve point $(x, y) = (h_1(\Psi, R_1), h_2(\Psi, R_1))$. After this calculation, a valid elliptic curve is found where this point is valid $y^2 = x^3 + ax + b$, where the value for $a$ is predefined. However, this does require extra encrypted information to be sent from $A$ to $B$, $[[R_1]]_\Psi, [[b]]_\Psi$. In a resource constrained environment, extra information sent by a sensor will require extra energy to be consumed by the sensor.

Using the RSA implementation [58] from the Deluge system and porting it to the mica2 mote system, and only using 160 bit exponents, the total number of cycles was found to be 147 879. However, when measuring the number of cycles by using the ECC implementation for sensors [107], including an implementation of the square root function, the total number that is obtained is 18 790 689. The key size was 160 bits, which is equivalent to 1024 bits in RSA. When moving to the ECC protocols, more secure keys are required. There is a significant number of extra cycles in a ECC implementation over the RSA implementation.

An examination of the memory of the application is examined, as shown in Table 5.5. The combination of .bss and .data segments use SRAM and the combination of .text and .data segments use ROM. The .text contains the machine instructions for the application. The .bss contains uninitialised global or static variables, and the .data section contains the initialised static variables.

<table>
<thead>
<tr>
<th>Memory</th>
<th>RSA Application</th>
<th>ECC Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
<td>1942</td>
<td>9720</td>
</tr>
<tr>
<td>RAM</td>
<td>177</td>
<td>859</td>
</tr>
<tr>
<td>.data</td>
<td>60</td>
<td>8</td>
</tr>
<tr>
<td>.bss</td>
<td>117</td>
<td>851</td>
</tr>
<tr>
<td>.text</td>
<td>1882</td>
<td>9712</td>
</tr>
</tbody>
</table>
5.6 Anderson–Lomas Protocol

In the literature, there is no protocol based on the EKE protocol that uses elliptic curves in an efficient deterministic method. The search is further extended to include other less-known protocols. One of them is the Anderson and Lomas protocol.

Protocol 5.7 Anderson–Lomas protocol

A and B both know $\Psi_1$ and $\Psi_2$.
A creates $t_A = g^{r_A}$, where $r_A$ is a random value.
B creates $t_B = g^{r_B}$, where $r_B$ is a random value.
$Z_{AB} = g^{r_A r_B}$

$M_1$ $A \rightarrow B$: $t_A$
$M_2$ $B \rightarrow A$: $t_B$
$M_3$ $A \rightarrow B$: $h_{\Psi}(K_{AB})$
$M_4$ $B \rightarrow A$: $h_{H(\Psi)}(K_{AB})$

As shown in Protocol 5.7, The first two messages are the Diffie–Hellman part of the protocol, whereas the second two messages authenticate that A sent $t_A$ and that B sent $t_B$. In traditional environments $\Psi_1$ and $\Psi_2$ are the same values, and would normally be a password shared by a client and server.

The function $h$ is a collisionful hash function. The function is supplied with two parameters, a key $k$ and a bit-string $x$. When function $h$ is supplied with $k$ and $x$ it is easy to compute $h_k(x)$. If function $h$ is supplied with $k$ and $h_k(x)$ it is hard to find a value $y$ such that $h_k(x) = h_k(y)$ but $x$ does not equal $y$. Given $x$ and $h_k(x)$ it is hard to compute $k$, though it is less hard to find $k'$ such that $h_k(x) = h_{k'}(x)$, where $k$ does not equal $k'$. The specific collisionful hash function used in the Anderson and Lomas protocol is:

$$h_k(x) = H(MAC_k(x) \mod 2^m, x)$$  \hspace{1cm} (5.2)

The value $m$ is a pre-defined value known by all parties and $H$ is a one-way hash function. It is easy to find many $k$ values that give the same output when $m$ is small. Anderson and Lomas suggest taking $m = n/2$ where $2^n$ is the size of the password space, allowing the protocol to use $\Psi_1$ and $\Psi_2$ as the key $h$ and so guess for either $\Psi_1$ or $\Psi_2$ can only be verified with probability $2^{-n/2}$.

An adversary can try to mount an active attack by posing as B and attempting to guess the message $M_3$; this will succeed with probability $2^{-n/2}$. Therefore, an active adversary can reduce the number of possible passwords to the square root of the initial number of passwords. Thus in traditional networks the protocol leaks a larger amount of information than the other protocols.

A defence against this type of attack is that if an invalid password check has been detected, the password needs to be changed. This is difficult in traditional networks and can
produce an effective denial-of-service attack. However, in an environment where the SEVs are continually changing, this protocol can be useful. The protocol was modified slightly to remove one of the messages, and to utilise elliptic curves, as shown in modified Protocol 5.8. Since the protocol does not encrypt the ephemeral values, encryption algorithms are no longer required.

**Protocol 5.8** Modified Anderson–Lomas protocol to use SEVs

A and B both know \( \Psi_1 \) and \( \Psi_2 \).

A creates \( t_A = r_A g \), where \( r_A \) is a random value.

B creates \( t_B = r_B g \), where \( r_B \) is a random value.

\[ Z_{AB} = r_A r_B g \]

\[ M_1 \quad A \rightarrow B : \quad t_A \]

\[ M_2 \quad B \rightarrow A : \quad t_B, h_{\Psi_1}(Z_{AB}) \]

\[ M_3 \quad A \rightarrow B : \quad h_{\Psi_2}(Z_{AB}) \]

The message \( M_1 \) is essentially the same as that found in the Anderson–Lomas protocol, except an elliptic curve point is sent. The message \( M_2 \) in the modified Anderson–Lomas protocol also contains an elliptic curve point; however, it also contains the authenticator instead of having a separate message. The message \( M_3 \) is the same as that found in the Anderson–Lomas protocol.

If SEVs do not change between several instantiations of this protocol, then the protocol suffers from an active attack; however, unlike the Anderson–Lomas protocol, the adversary can pose as \( A \), and thus the adversary does not need to wait for a sensor to send a request for a new key.

Table 5.6 supplies a quick guide of when to use some of the protocols described in the previous sections. If an ECC implementation currently exists on the sensor nodes, then the SPEKE or the modified Anderson–Lomas protocol can be used. However, these protocols require 160 bit ECC keys (since the RSA version require 1024 bit keys), and therefore, the ECC point that is transmitted is in total 320 bits in size. The EKE and the SEVN protocol only require 160 bit RSA keys.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>SEV Changes</th>
<th>Key Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>EKE</td>
<td>Infrequently</td>
<td>RSA</td>
</tr>
<tr>
<td>SEVN</td>
<td>Frequently</td>
<td>RSA</td>
</tr>
<tr>
<td>SPEKE</td>
<td>Infrequently</td>
<td>ECC</td>
</tr>
<tr>
<td>Modified Anderson–Lomas</td>
<td>Frequently</td>
<td>ECC</td>
</tr>
</tbody>
</table>

The values provided by the TOSSIM simulator (a part of the TinyOS installation) were used to obtain an indication of the power consumption when sending a message. In the calculations collision avoidance times are not taken into account. On the mica2 mote, the
cost of sending an extra 20 bytes is 28.1 microjoules. There is a substantial start-up cost for each message sent, and then there is an added cost for every bit that is sent.

5.6.1 Modified PPK Protocol

The PPK protocol [34] does not suffer from the attack described in the previous section. The PPK protocol can be used in sensor networks as a way to update small keys. The PPK protocol was implemented using elliptic curves, and the definition of the modified protocol is shown in Protocol 5.9. The $\Psi_1$ and $\Psi_2$ values in the protocol below are generated from two different SEVs. When using physiological data the low entropy key can change while the protocol is running.

**Protocol 5.9** Modified PPK protocol to use elliptic curve

<table>
<thead>
<tr>
<th>Shared Information: Generator $g$ of $G$ where $y^2 = x^3 + ax + b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
</tr>
<tr>
<td>$P_1 = r(f_1(A, B, \Psi_1))$</td>
</tr>
<tr>
<td>$x \in \mathbb{Z}_q$</td>
</tr>
<tr>
<td>$t_A = xg$</td>
</tr>
<tr>
<td>$m = t_A + P_1$</td>
</tr>
<tr>
<td>$\rightarrow$</td>
</tr>
<tr>
<td>$t_A = m - P_1$</td>
</tr>
<tr>
<td>$y \in \mathbb{Z}_q$</td>
</tr>
<tr>
<td>$t_B = yg$</td>
</tr>
<tr>
<td>$P_2 = r(f_1(B, A, \Psi_2))$</td>
</tr>
<tr>
<td>$t_B = m' - P_2$</td>
</tr>
<tr>
<td>$Z_{AB} = xt_B$</td>
</tr>
<tr>
<td>$m' = t_B + P_2$</td>
</tr>
<tr>
<td>$Z_{AB} = yt_A$</td>
</tr>
<tr>
<td>$K_{AB} = f_2(A, B, m, m', Z_{AB}, P_1)$</td>
</tr>
</tbody>
</table>

The modified PPK protocol contains two messages. The message $m$, sent by $A$, contains the first part of the Diffie–Hellman key agreement protocol. The authentication of message $m$ is obtained by adding an elliptic curve point, which is derived from a small key or in the new protocols an SEV. The message $m'$, sent by $B$, contains the second part of the Diffie–Hellman key agreement protocol. Once again, the authentication of message $m'$ is obtained by adding an elliptic curve point, which is also derived from a small key or SEV.

The most difficult part of this protocol is mapping $A, B, K_{AB}$ to a random point on the elliptic curve. The function $f_1$ is used to generate a point. A general procedure for this can be found in IEEE 1363 standard (Appendix A.11.1) [84]. The function $f_2$ is used to generate the new key $K_{AB}$, $f_2$ may be a one-way hash function.

5.7 Securing Sensors Measuring Different SEVs

In a heterogeneous sensor network, every sensor node may not be able to measure the same environmental phenomenon. Poon et al. [130] suggested that all nodes will need to
have the same sensor attached to them. However, adding the same sensor to every node may be cost prohibitive. Another solution is to use appropriate protocols, where a node that can sense both phenomena (or is sent information about both phenomena) can be used to translate the messages. This section will show that three party password protocols can establish keys between two sensors that measure different environmental data.

A cornerstone to wireless sensor network security is key establishment protocols. In traditional networks, when using symmetric key cryptography, if two entities sharing no previous secret want to communicate securely with each other, they generally do so with the assistance of a third party [33]. There is a class of problems in WSNs where one or more sensor nodes act as a trusted intermediary to facilitate key establishment [42, 127]. Typically, the trusted intermediary provides an authentication service that distributes a secure session key to the sensor nodes. The issues with using a sensor node as the trusted third party are:

- The trusted intermediary may not have any established keys.
- Key sizes in sensors nodes are not large.
- Sensor networks may only require authenticated messages, so may not have access to an encryption algorithm.

Several existing password protocols [105, 163] handle small keys in the three party scenarios. The Steiner et al. protocol [163] has a number of flaws [52, 104], which makes the protocol unsuitable for the sensor environment. The Lin et al. protocol [105] has no known attack; however, it requires seven separate messages. Each message a sensor node sends or receives requires extra energy which the sensor may not have. Abdalla et al. [2] supplied the first security proof to the three party scenarios. However, the number of messages in the protocol is still large; the protocol is defined below in Protocol 5.10.

**Protocol 5.10** Abdalla three party protocol

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>A $\leftrightarrow$ S: Use a two party password protocol to establish a session key $K_{AS}$.</td>
</tr>
<tr>
<td>P2</td>
<td>B $\leftrightarrow$ S: Use a two party password protocol to establish a session key $K_{BS}$.</td>
</tr>
<tr>
<td>P3</td>
<td>A $\leftrightarrow$ S: Use a two party key distribution protocol to create a MAC key $K_m$.</td>
</tr>
<tr>
<td>P4</td>
<td>B $\leftrightarrow$ S: Use a two party key distribution protocol to create a MAC key $K_m$.</td>
</tr>
<tr>
<td>M1</td>
<td>A $\rightarrow$ B: $g^x, [g^x, B, A]_{K_m}$</td>
</tr>
<tr>
<td>M2</td>
<td>B $\rightarrow$ A: $g^y, [g^y, B, A]_{K_m}$</td>
</tr>
</tbody>
</table>

The Abdalla three party protocol is a combination of several key establishment protocols. First a session key $K_{AS}$ is created, then the session key $K_{BS}$ is created. The key $K_m$ is then created, so that both $A$ and $B$ has access to it. The new key $K_m$ is then used to authenticate the Diffie–Hellman exchange.

In sensor networks, communication costs are expensive, which makes the above protocols undesirable. A new protocol is proposed by combining two password protocols with a KDC protocol.
An investigation of the available KDC protocols is performed with the intention of extending one of the protocols into a suitable three party password protocol. Boyd and Mathuria have surveyed authentication and key establishment protocols [33] in traditional networks. However, the protocols have not been extensively analysed for WSN environments.

Boyd and Mathuria have listed a total of 22 server-based key establishment protocols [33]. For the analysis a reduction of the number of protocols is performed to get the protocols to a manageable size of seven. Keeping sensor environments in mind, the following steps were used to reduce the number of protocols. Protocols requiring sensor nodes that store old messages to prevent replay attacks have been removed. If an optimised version of the protocol exists, it will be looked at in preference to the older protocol. The requirement of a flexible protocol for most situations helped removed the protocols relying on timestamps, as not all sensor environments are guaranteed to have secure time synchronisation. Another requirement is to minimise the amount of communication, any protocols requiring five or more messages were removed, leaving the protocols that send only four messages, as shown in Table 5.7.

Table 5.7: Comparison of Existing Traditional Network Single Server Protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Properties for the Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bauer et al. [18]</td>
<td>S</td>
</tr>
<tr>
<td>Otway and Rees [124]</td>
<td>S</td>
</tr>
<tr>
<td>Yahalom Burrows et al. [38]</td>
<td>S</td>
</tr>
<tr>
<td>Bellare and Rogaway [21]</td>
<td>S</td>
</tr>
<tr>
<td>Abadi and Needham [1]</td>
<td>S</td>
</tr>
<tr>
<td>Mechanism 13 ISO11770–2 [87]</td>
<td>B</td>
</tr>
<tr>
<td>Boyd Four-Pass [30]</td>
<td>S/A/B</td>
</tr>
</tbody>
</table>

The remaining protocols differed in who controlled the creation of the session key $K_{AB}$, and which sensors confirm that the other has the session key. The Boyd protocol has the most features where the key is generated from information from nodes $A$, $B$ and $S$, as well as having key confirmation between nodes $A$ and $B$. The next one is the Yahalom protocol, which has key confirmation for node $B$, with the node $S$ generating the session key. The only other protocol that does not have node $S$ generating the session key is Mechanism 13 found in the ISO standard 11770–2.

The number of messages sent in sensor networks does not give a true indication of the communication costs. The size of the messages and thus the number of bytes should also be examined. Table 5.8 shows the amount of bytes sent by each protocol. It is assumed that $A$, $B$ and $S$ communicate with each other over the sensor network. The size of a node address is one byte, in TinyOS the node address is 2 bytes allowing for 64K nodes to be in
the sensor network. A one byte address allows 256 nodes to be addressed, which should be a large enough address space for sensors on the body. The size of the nonce is one byte, allowing the protocol to be rerun 256 times before a replay attack is possible. The size of the session key is 64 bits, which is the default value in TinyOS.

Table 5.8: Number of Bytes Sent by Each Protocol

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Bytes Sent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bauer et al. [18]</td>
<td>48</td>
</tr>
<tr>
<td>Otway and Rees [124]</td>
<td>67</td>
</tr>
<tr>
<td>Yahalom Burrows et al. [38]</td>
<td>56</td>
</tr>
<tr>
<td>Bellare and Rogaway [21]</td>
<td>30</td>
</tr>
<tr>
<td>Abadi and Needham [1]</td>
<td>52</td>
</tr>
<tr>
<td>Mechanism 13 ISO11770–2 [87]</td>
<td>49</td>
</tr>
<tr>
<td>Boyd Four-Pass [30]</td>
<td>43</td>
</tr>
</tbody>
</table>

From the remaining seven protocols, the protocol with the least number of bytes sent is the Bellare–Rogaway protocol [21]. The Bellare–Rogaway protocol also has the feature that the node $S$ can get a message from both sensor nodes $A$ and $B$ (either directly or indirectly). Also node $S$ returns a message to the sensor nodes $A$ and $B$. The communication structure of the Bellare–Rogaway protocol allows a Diffie–Hellman based protocol to be piggy-backed upon the three party protocol.

A comparison of the proposed protocols is performed with existing KDC WSN protocols, such as SPINS [127] and PIKE [42]. Other KDC WSN protocols, such as TUTWSN KDC protocol [72], do not meet the criteria of having less than five messages and are therefore not considered.

The KDC protocol described in PIKE suffers from a replay attack, since the messages do not contain nonces or timestamps. Another problem with the PIKE protocol is the lack of session identifiers. This has the effect of only having one running instance of the protocol at a time. The other protocols described in this chapter have nonces, which can be used as session identifiers [49] so they do not suffer from this issue. Because of the major limitations of the KDC protocol described in PIKE, the security analysis will not be investigated any further here.

The SPINS KDC protocol contains four messages, and has a total of 34 bytes sent. The SPINS KDC protocol is similar to the Bellare–Rogaway protocol, except there is an extra MAC in the message $M2$. The Bellare–Rogaway protocol is proven to be secure and the need for the extra MAC is unnecessary. Therefore, due to the similarity that the SPINS KDC protocol has with the Bellare–Rogaway protocol, the security analysis of the SPINS KDC protocol will not be investigated any further either.

The number of messages, the size of the messages, and the communication structure makes the Bellare–Rogaway protocol an ideal protocol to convert into a three party pass-
word protocol. The Bellare–Rogaway Three Party Key Distribution Protocol (3PKDP) [21] is shown in Protocol 5.11.

**Protocol 5.11 Bellare–Rogaway 3PKDP**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>A → B : A, N_A</td>
<td>M3</td>
<td>S → A : [([K_{AB}]<em>{K</em>{AS}}), [A, B, N_A, [([K_{AB}]<em>{K</em>{AS}})]<em>{K</em>{AS}}]<em>{K</em>{AS}}]</td>
<td></td>
</tr>
<tr>
<td>M2</td>
<td>B → S : A, B, N_B, N_B</td>
<td>M4</td>
<td>S → B : [([K_{AB}]<em>{K</em>{BS}}), [A, B, N_B, [([K_{AB}]<em>{K</em>{BS}})]<em>{K</em>{BS}}]<em>{K</em>{BS}}]</td>
<td></td>
</tr>
</tbody>
</table>

The Bellare–Rogaway protocol contains four messages. The sensor node A sends the message M1 to sensor node B. The message M1 contains the source address, since TinyOS does not send the source address in its packet header. The message also contains a nonce generated by A; the nonce can either be a counter or random number. The sensor node B sends the message M2 to sensor node S. The message M2 contains the contents of message M1 as well as the address of sensor node B and a nonce generated by sensor node B. Sensor node S then sends an encrypted key $K_{AB}$ to both of the nodes A and B.

Unlike in the SEVN protocol the encryption in the Bellare–Rogaway protocol cannot be a simple exclusive-or function. If the key $K_{AB}$ becomes compromised, then the keys $K_{AS}$ and $K_{BS}$ can also be compromised. Thus, if this protocol is rerun, any session keys between the nodes A and B will become compromised. Whereas, if a block cipher is used to encrypt the session key $K_{AB}$, then if the key $K_{AB}$ becomes compromised, the keys $K_{AS}$ and $K_{BS}$ will not become compromised.

### 5.7.1 Proposed Three Party Protocol

This section will show that when the Bellare–Rogaway protocol is converted into a three party password protocol, the restriction on the encryption cipher is relaxed. The exponent is a random value defined as $r_A \in_R \mathbb{Z}_p$. The size of $p$ is 160 bits. The Diffie–Hellman ephemeral value is defined as $t_A = g^{r_A}$. Four ephemeral values are created for the three party protocol, each created with different random exponents of $\mathbb{Z}_p$. The value $t_{AS}$ is the ephemeral value created by sensor node A to be sent to sensor node S (indirectly by sending it to sensor node B). The value $t_{SA}$ is the ephemeral value created by sensor node S to be sent to sensor node A. The two other ephemeral values are $t_{BS}$ and $t_{SB}$. The value $t_{BS}$ is the ephemeral value created by sensor node B to be sent to sensor node S. Whereas, the ephemeral value $t_{SB}$ is created by sensor node S to be sent to sensor node B.

The ephemeral values are encrypted by the SEV values. The SEV values for each ephemeral value may be different, since the SEV may change over time. However, the protocol does not limit the SEV values to always be different; the SEV value used in each construct may be the same. The following constructs are used when defining the three party password protocol:
5.7. SECURING SENSORS MEASURING DIFFERENT SEVS

\[
m_{AS} = [[t_{AS}]]_{\psi_1} \\
m'_{AS} = [[t_{SA}]]_{\psi_2} \\
m_{BS} = [[t_{BS}]]_{\psi_3} \\
m'_{BS} = [[t_{SB}]]_{\psi_4}
\]

(5.3)

The Combined Symmetric Password Key (CSPK) protocol is shown in Protocol 5.12.

**Protocol 5.12** CSPK — Combined Symmetric Password Key protocol

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(M1)</td>
<td>(A \rightarrow B)</td>
<td>(A, m_{AS})</td>
</tr>
<tr>
<td>(M2)</td>
<td>(B \rightarrow S)</td>
<td>(A, B, m_{AS}, m_{BS})</td>
</tr>
<tr>
<td>(M3)</td>
<td>(S \rightarrow A)</td>
<td>(m'<em>{AS}, K</em>{AS} \oplus K_{AB}, [A, B, K_{AS} \oplus K_{AB}]<em>{K</em>{AS}})</td>
</tr>
<tr>
<td>(M4)</td>
<td>(S \rightarrow B)</td>
<td>(m'<em>{BS}, K</em>{BS} \oplus K_{AB}, [A, B, K_{BS} \oplus K_{AB}]<em>{K</em>{BS}})</td>
</tr>
</tbody>
</table>

The CSPK protocol contains a total of four messages and is based on the provably secure Bellare–Rogaway 3PKDP [21]. Additional data \((m_{AS}, m_{BS}, m'_{AS}, m'_{BS})\) has been added to each message from the Bellare–Rogaway 3PKDP, so that an ephemeral key can be constructed between \(A\) and \(S\), and another ephemeral key between \(B\) and \(S\). When node \(S\) receives message \(M2\), node \(S\) can create the ephemeral keys \(K_{AS}\) and \(K_{BS}\) since message \(M2\) contains both \(m_{AS}\) and \(m_{BS}\). It can use the ephemeral keys to encrypt \(K_{AB}\) and create the MAC. When \(S\) sends the message to \(A\), it can calculate the session key \(K_{AS}\), and when \(S\) sends the message to \(B\), it can calculate the session key \(K_{BS}\). Once the protocol is finished, these session keys can be discarded. Thus, if ever \(S\) was compromised, then the value \(K_{AB}\) will not be compromised.

Since the keys \(K_{AS}\) and \(K_{BS}\) are ephemeral keys, this protocol can use exclusive-or function as the encryption algorithm. If the key \(K_{AB}\) does become compromised, then since the keys \(K_{AS}\) and \(K_{BS}\) are only used to establish the keys between the nodes \(A\) and \(B\), the protocol will not give away any extra information outside of the protocol.

When the SEV changes every time that the protocol is run, the protocol can have the extra change of removal of the nonces. This change does limit the protocol to environments where the SEV changes between protocol runs. If the SEV does not change, then the nonces will need to be added back into the protocol.

Previously, the defence against the replay attack was through the use of the nonces. In the CSPK protocol the defence against the replay attack is the creation of ephemeral keys \(K_{AS}\) and \(K_{BS}\). If an adversary replays \(M1\), then \(S\) will return a different \(m'_{AS}\) and thus a new ephemeral key \(K_{AS}\) is be created and a new message \(M3\) will be returned (giving the adversary no new information). If an adversary replays the message \(M2\), then \(S\) will return a different \(m'_{BS}\) and thus a new ephemeral key \(K_{BS}\) is created. The adversary will gain no new information. If an adversary replays the messages \(M3\) or \(M4\), then the sensor
nodes $A$ or $B$ will detect the messages as invalid, since sensor nodes $A$ and $B$ expected new ephemeral keys.

### 5.7.2 Three Party Protocol with One Password Protocol

The Boyd three party protocol [30] has been extended for the sensor network in the previous chapter. The Boyd protocol has the advantage of key authentication, key freshness and key confirmation in only four messages. The nodes $A$ and $B$ exclusively share a secret ($K_{AS}$ and $K_{BS}$ respectively) with the trusted authentication server $S$. By executing the following protocol, $A$ and $B$ intend to establish a session key $K_{AB}$. One of the properties of the key established between $A$ and $B$ is the use of information from $S$, $A$, and $B$ to create the session key. The protocol has been modified so that it better suits the wireless sensor environment. The need for encryption has been removed by using the same technique as Janson and Tsudik [89]. The nonces have also been used as session identifiers so they do not suffer from security attacks across multiple sessions [49]. The final three party protocol proposed for wireless sensor networks is the Enhanced No Encryption Key Agreement (ENEKA) protocol as shown in Protocol 5.13.

**Protocol 5.13**  ENEKA — Enhanced No Encryption Key Agreement protocol

$M1 \quad A \rightarrow S : \quad A, B, N_A$

$M2 \quad S \rightarrow B : \quad A, N_A, \Theta_B, \Phi_B \oplus K_S, \Theta_A, \Phi_A \oplus K_S$

$M3 \quad B \rightarrow A : \quad [N_A]_{K_{AB}}, N_B, \Theta_A, \Phi_A \oplus K_S$

$M4 \quad A \rightarrow B : \quad [N_B]_{K_{AB}}$

The following constructs are used:

\[
\begin{align*}
\Theta_A &= [A, B, K_S]_{K_{AS}} \\
\Phi_A &= [[\Theta_A]]_{K_{AS}} \\
\Theta_B &= [A, B, K_S]_{K_{BS}} \\
\Phi_B &= [[\Theta_B]]_{K_{BS}}
\end{align*}
\] 

(5.4)

The $\Phi$ values can either be created from an encryption algorithm or from a MAC. In cases like CBC–MAC, where the algorithm uses an underlying encryption algorithm, it may be more efficient to use encryption. In other cases where there is no encryption algorithm available (for instance, only HMAC–MD5 or hardware support) then the $\Phi$ values can be created by using the MAC. The size of $\Phi$ should be equal to or greater than the size of the keys. By default TinySEC has a 32 bit MAC; however, the CBC–MAC can create a variable size MAC. If key confirmation functionality is not required then messages $M3$ and $M4$ can be removed.

An attractive feature of this protocol is that $K_{AB}$ is generated from information from $A$, $B$ and $S$, as shown in Equation (5.5). The nonces are used to guarantee the key is fresh.
5.7. SECURING SENSORS MEASURING DIFFERENT SEVS

The nonces do not need to be random, although $K_S$ should be a good random number.

$$K_{AB} = [N_A, N_B]_{K_S} \quad (5.5)$$

5.7.2.1 Attacks and Limitations in the Boyd Protocol

One of the problems with the Boyd protocol is that it is prone to the replay attack. It cannot be ruled out that $K_S$ can become compromised, such as from a brute force attack. If a new $K_S$ needs to be created, the natural thought will be to run the proposed protocol from the start. However, there is an attack where an adversary can replay a portion of a previous message, so that the session key is still $K_S$. Note that there is no way to guarantee that the encryption of $A, B, K_S$ is an old value or a new value. This type of attack can be classified as a replay attack. If the keys between the nodes and the KDCs are updated more frequently than $K_S$, then the proposed protocols can be run from the start without fear of a replay attack. The different keys between the nodes and the KDCs stop any adversary performing a replay attack. However, this may not be possible in all wireless sensor networks. Boyd’s defence against the replay attack relies on a revocation list being available to all parties [30], but having a revocation list infrastructure may not be feasible in a low bandwidth, low energy, and low computational environment. The previous chapter proposed the use of multiple servers. However, that only solves the problem of $K_S$ becoming compromised from the server becoming compromised. Multiple servers do not solve the problem if the key $K_S$ becomes compromised by the other means, such as a brute force attack.

5.7.3 Proposed Protocol using the Boyd Protocol

In a heterogeneous environment, some nodes may be able to sense different cryptographically strong values. In this section a protocol will be proposed where one sensor node can sense a cryptographically strong SEV (labelled $Ψ_{BS}$) and the other sensor can sense a weak $Ψ$ (labelled $Ψ_1$ and $Ψ_2$). The proposed protocol is based on the Boyd three party protocol [30], which has been proposed for the sensor network in Chapter 4. The Boyd protocol has the advantage of key authentication, key freshness and key confirmation in only four messages. In the proposed Password No Encryption Key Agreement (PNEKA) protocol, as shown in Protocol 5.14, the Diffie–Hellman part of the protocol is defined in Equation (5.6), where $r_A$ and $r_S$ are random values.

$$m_{AS} = [[t_{AS}]]_{Ψ_1}$$

$$m'_{AS} = [[t_{SA}]]_{Ψ_2}$$

$$K_{AS} = t_{AS}^{r_S} = t_{SA}^{r_A} \quad (5.6)$$
CHAPTER 5. ESTABLISHING INITIAL KEYS

The symmetric key constructs of the PNEKA protocol is defined in Equation (5.7). The \( \Phi \) values can either be created from an encryption algorithm or from a MAC. In cases like CBC–MAC, where the algorithm uses an underlying encryption algorithm, it may be more efficient to use encryption. In other cases where there is no encryption algorithm available (for instance, only HMAC–MD5 or hardware support) then the \( \Phi \) values can be created by using the MAC. The size of \( \Phi \) should be equal to or greater than the size of the keys. By default TinySEC has a 32 bit MAC; however, the CBC–MAC used in TinySEC can create a variable size MAC.

\[
\begin{align*}
\Theta_A &= [A, B, K_S]_{K_{AS}} \\
\Phi_A &= ([\Theta_A])_{K_{AS}} \\
\Theta_B &= [A, B, K_S]_{\Psi_{BS}} \\
\Phi_B &= ([\Theta_B])_{\Psi_{BS}}
\end{align*}
\]

(5.7)

Protocol 5.14 PNEKA — Password No Encryption Key Agreement protocol

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Message 1:</th>
<th>Message 2:</th>
<th>Message 3:</th>
<th>Message 4:</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>( A \rightarrow S ): ( A, B, N_A, m_{AS} )</td>
<td>( S \rightarrow B ): ( A, N_A, \Theta_B, \Phi_B \oplus K_S, \Theta_A, \Phi_A \oplus K_S, m'_{AS} )</td>
<td>( B \rightarrow A ): ( [N_A]<em>{K</em>{AB}}, N_B, \Theta_A, \Phi_A \oplus K_S, m'_{AS} )</td>
<td>( A \rightarrow B ): ( [N_B]<em>{K</em>{AB}} )</td>
</tr>
</tbody>
</table>

Unlike the CSPK protocol where node \( S \) receives a message created by both the nodes \( A \) and \( B \), in the PNEKA protocol the node \( S \) can only receive data from node \( A \). In cases where the only weak SEV is between the nodes \( A \) and \( S \), the PNEKA protocol can be used. The protocol contains four messages; the messages are based on the Boyd three party protocol designed for sensor networks as described in the previous chapter. Additional data \((m_{AS}, m'_{AS})\) has been added to each message, so that an ephemeral key can be constructed between \( A \) and \( S \). The key between \( B \) and \( S \) can be based on an SEV, which both \( B \) and \( S \) can obtain or sense.

As with Boyd protocol the proposed protocol also has the feature that \( K_{AB} \) is generated from information from \( A, B \) and \( S \), as shown in Equation (5.5). As with the Boyd protocol the nonces are used to guarantee that the key is fresh, and the nonces do not need to be random, although \( K_S \) should be a good random number.

One of the problems with the Boyd protocol is that it is prone to the replay attack. Boyd’s defence against the replay attack relies on a revocation list being available to all parties, but having a revocation list infrastructure may not be feasible in a low bandwidth, low energy, and low computational environment. The \( \Theta \) and \( \Phi \) values could be replayed by an adversary. However, in the PNEKA protocol new ephemeral keys \( K_{AS} \) and \( K_{BS} \) are created with each running of the protocol. The creation of the new ephemeral keys stops any replay attacks.
If key confirmation functionality is not required then parts of message \( M3 \) and all of message \( M4 \) can be removed. There is a case where the leader node and the base station can communicate without the need for environmental values. Here is a three party protocol, where a sensor node, a leader node, and a base station want to communicate. For instance, the base station may want to directly communicate with a particular sensor node. However, the sensor node does not have a key with the base station.

### 5.7.4 Proposed Three Party Protocol with No Password Protocol

Perrig et al. [127] assumed that communicating parties, such as \( A \) and \( B \), know each other’s counter values, which can be used as nonces. So the nonces do not need to be added to each message. However, if messages get lost, the shared counter state can become inconsistent. Protocols were suggested to synchronise the counter between the nodes. However, in a resource constrained environment this is considered to be undesirable.

Another solution is proposed that can be used in wireless sensor networks, in particular body area sensor networks. The protocol is similar to the Boyd three party protocol for sensors. However, the constructs \( \Theta_A \) and \( \Theta_B \) are changed. Also, a new term is defined, Environment Value (EV). Unlike the SEVs, an EV is environmental data without the need for it to be a secret. The No Encryption Key Agreement with Environment Values (NEKAEV) protocol was created, and is shown in Protocol 5.15.

#### Protocol 5.15 NEKAEV — No Encryption Key Agreement with Environment Values

\[
\begin{align*}
\Theta_A &= [A, B, K_{S}, EV_{AS}, K_{AS}] \\
\Theta_B &= [A, B, K_{S}, EV_{BS}, K_{AS}]
\end{align*}
\]

Run ENEKA (Protocol 5.13).

The symbol \( EV_{AS} \) is a shared EV between the sensors \( A \) and \( S \). Whereas, the symbol \( EV_{BS} \) is a shared EV between the sensors \( B \) and \( S \). There is no requirement that \( EV_{AS} \) and \( EV_{BS} \) cannot be the same measured phenomena. However, the constructs \( \Theta_A \) and \( \Theta_B \) are designed so that different physiological phenomena may be used between \( A \) and \( S \), and \( B \) and \( S \).

Sensors can use environmental data (if the data does not regularly repeat) as the nonce values. So instead of sending nonces, the protocol will use the shared environmental values as the nonce values.

Since \( A \) and \( S \) both share \( EV_{AS} \) and \( K_{AS} \), as long as \( EV_{AS} \) is random (such as HRV or IPI in the body) then \( \Theta_A \) cannot be replayed when the protocol is rerun. The same applies for \( B \) and \( S \). If a \( EV_{AS} \) has \( n \)-bits of randomness, then by the birthday paradox, an expectation (probabilistically) will be to see the first repetition for \( EV_{AS} \) after roughly \( 2^{n/2} \) runs of the protocol. The same applies for \( EV_{BS} \). However, if the EVs can also be considered to be SEVs, then the adversary will not be able to know when the EV has repeated.
The EVs can also be used to update the keys after the running of the Boyd three party protocol for sensors. At a specified time, a session key can be generated between the sensors A and B, by using the Equation (5.8), assuming that both sensors A and B can read the same EV.

\[ K_{AB} = [EV_{AB}]_K \] (5.8)

However, if the sensors cannot read the same environmental value, then a counter can be used in place of the \( EV_{AB} \), or nonces will need to be sent between the sensors. The previous chapter showed that a two party entity authentication protocol can be used to create the session key when using the Boyd three party protocol. The example of the two party entity authentication protocol was the provably secure Bellare–Rogaway MAP1 protocol [20], which consists of a total of three messages. Instead the simpler Boyd Two-Pass (BTP) protocol [31] can be used. It contains a total of two messages as shown in Protocol 5.16.

**Protocol 5.16 BTP — Boyd Two-Pass protocol**

\[
\begin{align*}
M1 & \quad A \rightarrow B : \quad A, N_A \\
M2 & \quad B \rightarrow A : \quad B, N_B
\end{align*}
\]

BTP protocol allows both of the sensors A and B to supply information for the session key \( K_{AB} \) as shown in Equation (5.9).

\[ K_{AB} = [N_A, N_B]_K \] (5.9)

Another solution is that only one message \( M1 \) is sent, changing the Equation (5.9) to be \( K_{AB} = [N_A]_K \). The sensor B supplies less input into the session key; however, sensor B did originally supply input into creating the key \( K \).

### 5.8 A Group Protocol

If several entities sharing no previous secret need to create a shared group key, a group key establishment protocol can be utilised, such as the Burmester and Desmedt protocol as defined in Protocol 5.17 [36]. The protocol was originally created without a formal proof, but subsequent work has shown the protocol to be provably secure [96]. The protocol provides both forward and backwards secrecy. Where an adversary who knows a set of group keys cannot derive any new group keys or an adversary who knows a set of group keys cannot find any earlier group keys. The Burmester and Desmedt protocol uses a cyclic function to define the shared key. The protocol requires that the sensors are logically arranged in a ring so that \( S_1 = S_{m+1} \), where \( m \) is the total number of sensors.
Protocol 5.17 Burmester–Desmedt generalised Diffie–Hellman with broadcasts

<table>
<thead>
<tr>
<th>$S_{i-1}$</th>
<th>$S_i$</th>
<th>$S_{i+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round 1:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_i$</td>
<td>$r_i \in_R \mathbb{Z}_p$</td>
<td>$t_i$</td>
</tr>
<tr>
<td>$t_i = g^{r_i}$</td>
<td>$X_i = (t_{i+1}/t_{i-1})^{r_i}$</td>
<td>$Z_{i-1,i} = t_i^{r_i}$</td>
</tr>
<tr>
<td>Round 2:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_i$ broadcasts $X_i$ to all parties</td>
<td>$S_i$ calculates $Z$</td>
<td></td>
</tr>
</tbody>
</table>

During Round 1 each adjacent pair of sensors, $S_i$ and $S_{i+1}$, performs a basic Diffie–Hellman key exchange. The sensor $S_i$ calculates the ratio, $X_i = (t_{i+1}/t_{i-1})^{r_i}$, from its two secrets from the adjacent sensors. In Round 2 each sensor broadcasts its $X_i$ value. Once all the sensors have broadcast their values, every sensor can calculate the shared secret $Z = (Z_{i-1,i})^{mX_i^{m-1}X_{i+1}^{m-2} \ldots X_{i-2}}$.

The protocol has no authentication mechanism and by itself cannot be used in a sensor environment. When using this protocol with SEVs, the difficulties found in this protocol included:

- There is no authentication or key confirmation mechanism.
- Using weak keys to secure the protocol is non-trivial.
- There are $m + 2$ exponents that are needed to be calculated per sensor.

Since the protocol has no authentication mechanism, an adversary can easily join the protocol and discover the group key. This section will show how SEVs can be used to authenticate all the participants in a group protocol, as long as every participant has access to the physiological values of the body (as an example) in real time. Key confirmation is also important to insure that all the participants in the group protocol obtained the newly created group key. This section will also show how SEVs can also be used as nonces.

Also, it is non-trivial using small keys (such as SEVs) to encrypt the communication. A security weakness occurs if the same weak key is used multiple times in the protocol. The same environmental data may be used over multiple protocol runs to conserve energy, or may be the environmental data stays constant over a period of time. When environmental data is used to encrypt $t_i$, an adversary can use brute force to decrypt $[[t_i]]_\Psi$ over all the possible values of $\Psi$ and check if the candidate results are valid. If they are not then that particular SEV can be removed from the list of valid environmental values. If the same SEV is used multiple times, then the SEV can become compromised, and the adversary can join in the protocol and obtain the group key.

Encryption of data using a weak key introduces a brute force style attack in this protocol. An adversary guessing the SEV values can attempt to decrypt $[[t_i]]_\Psi$ and examine
whether the resulting plaintext is a valid Diffie–Hellman ephemeral value. For instance, if the decryption with a candidate \( \Psi \) results in a string whose value is greater than \( p \), then that candidate is invalid.

Boyd and Mathuria [33] suggested an alternative method to calculate a shared secret when using a cyclic function. The method is to define \( X_i = Z_{i,i+1} - Z_{i-1,i} \), and the way to calculate the shared secret is defined as Equation (5.10).

\[
Z = mZ_{i-1,i} + (m-1)X_i + (m-2)X_{i+1} + \ldots + X_{i-2} \\
= mZ_{i-1,i} + (m-1)(Z_{i,i+1} - Z_{i-1,i}) + (m-2)(Z_{i+1,i+2} - Z_{i,i+1}) + \ldots + Z_{i-2,i-1} - Z_{i-3,i-2} \\
= Z_{1,2} + Z_{2,3} + \ldots + Z_{m,1} \\
= g^{r_1r_2} + g^{r_2r_3} + \ldots + g^{r_mr_1} \tag{5.10}
\]

However, there is a redundancy with Equation (5.10). The following is true \( X_1 + X_2 + \ldots + X_n = 0 \). This redundancy enables an adversary to make the protocol vulnerable to an attack by guessing the SEVs off-line and verifying whether the addition of decrypted values in the second round communication is zero. If so, the adversary’s guess for SEVs is correct. Every participant needs to complete round 2 for this attack to be successful. Using SEVs to only encrypt the second phase does not yield a secure protocol and one can mount an off-line attack.

### 5.8.1 Proposed Group Protocol using SEVs

This section will describe in detail the proposed group key establishment protocol. It is based on the provably secure Burmester and Desmedt protocol. Authentication and key confirmation has been added to this protocol using SEVs. Assumption is made that the protocol has \( n \) sensors, where each sensor is able to sense the same environmental values as the other sensors in the system. The **SEV Group Key (SEVGK)** protocol is described in Protocol 5.18.

In the first round, each sensor \( S_i \) chooses a random time \( T_i \) where \( (T_i > \text{currentTime}) \). The time will be used to notify all the participants at what time the sensor will be measuring its SEV. The time is set in the future so that an adversary cannot pick a time in the past, where they have calculated an old SEV. Each sensor will then broadcast out a message \( S_i, T_i \) informing all the other sensors at what stage they will calculate the SEV.

In the second round, each sensor will sense its environment and store the SEV \( \Psi_i \) values for each sensor that is involved in the protocol. If two sensors are measuring the same SEV, the sensor with the larger id value will be given the next available time slot in the list. Ensuring that \( \Psi_i \neq \Psi_j \forall i, j \).

In the third round, the Diffie–Hellman proportion of the protocol is performed. How-
### Protocol 5.18 SEVGK — SEV Group Key protocol

| Round 1: |  |
|----------|  |
| $S_{i-1}$ | $S_i$ | $S_{i+1}$ |
| $S_i$ chooses a time $T_i$ | $S_i$ broadcasts $S_i, T_i$ |  |

| Round 2: |  |
|----------|  |
| $S_i$ senses $V_1, \ldots, V_n$ | $S_i$ generates $\Psi_1, \ldots, \Psi_n$ |  |

| Round 3: |  |
|----------|  |
| $r_i \in \mathbb{Z}_p$ | $t_i = g^{r_i}$ |
| $t^*_i = t_i \oplus \Psi_i$ | $Z_{i-1,i} = t_{i-1}^*$ |
| $Z_{i+1,i} = t_{i+1}^*$ | $X_i = Z_{i+1,i} - Z_{i-1,i}$ |

| Round 4: |  |
|----------|  |
| $S_i$ broadcasts $X_i \oplus \Psi_i$ except $S_n$ | $S_i$ calculates $X_n$ and then $Z$ |  |

| Round 5: |  |
|----------|  |
| $S_i$ broadcasts $\text{ENC}_Z(S_i, \Psi_i)$ |  |

ever, before sending out $t_i$, an encryption is performed using $\Psi_i$ as the key. Since each SEV is different, a partition attack is not possible.

In the fourth round each sensor sends $X_i \oplus \Psi_i$, except for $S_n$. After all the messages have been broadcast, all the sensors will be able to calculate $X_n$ from the equation $X_n = -(X_1 + X_2 + \ldots + X_{n-1})$. The reason to have the value $X_n$ calculated by each sensor is to remove the redundancy found in Equation (5.10).

The fifth round verifies that each sensor has successfully calculated the new group key. It is important to add key confirmation at the end of this protocol, since an adversary may attempt a man-in-the-middle attack. This is done by each sensor encrypting the values $S_i, \Psi_i$ with the group key $Z$, and then broadcasting the encrypted values to each of the other sensors in the group.

### 5.8.2 Comparison with Public Key Infrastructure

A Public Key Infrastructure (PKI) [4, 166] binds public keys with their user identities by a Certificate Authority (CA). The identity of the user must be unique within each CA domain. A Validation Authority (VA), which may be a trusted third-party, can provide this information on behalf of the CA. The binding of the user identity and the public keys is established through the registration and issuance process. The registration and issuance process may be carried out by automatically at a CA, or under human supervision, where under human supervision has the higher level assurance. The PKI infrastructure
that ensures the binding of the public key with the user identity is called the Registration Authority (RA). The RA can also ensure non-repudiation for the issuance of the public keys to the user.

Other components in PKI also include:

- A central directory.
- A certificate management system.
- A certificate policy.

The password protocols do not require the extensive infrastructure found in a PKI. A complete PKI in a body sensor network where there may not always be access to a central system, may be too resource intensive for small devices. If the storage and computation costs were minimised, there is still the large communication overhead that can occur when a new session key is established.

5.9 Analysis and Implementation

This section will first describe a performance comparison between the Lin protocol and the CSPK protocol. The values for the Lin protocol were obtained directly from their publication [105]. The number of messages sent by each sensor node is also included. The Lin protocol is shown in Protocol 5.19.

### Protocol 5.19 Lin Protocol

<table>
<thead>
<tr>
<th>Step</th>
<th>Message</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>( A \rightarrow S ): ( A, B )</td>
<td></td>
</tr>
<tr>
<td>M2</td>
<td>( S \rightarrow A ): ( \left[ g^{N_{S1}} \right]<em>{K</em>{SA}}, \left[ g^{N_{S2}} \right]<em>{K</em>{SA}} )</td>
<td></td>
</tr>
<tr>
<td>M3</td>
<td>( A \rightarrow B ): ( A, R_A, f_{K_A}(A, B, g^{N_{S1}}), R_B, f_{K_B}(A, B, g^{N_{S2}}) )</td>
<td></td>
</tr>
<tr>
<td>M4</td>
<td>( B \rightarrow S ): ( R_A, f_{K_A}(A, B, g^{N_{S1}}), R_B, f_{K_B}(A, B, g^{N_{S2}}) )</td>
<td></td>
</tr>
<tr>
<td>M5</td>
<td>( S \rightarrow B ): ( f_{K_B}(A, B, R_A, R_B), f_{K_A}(A, B, R_A, R_B) )</td>
<td></td>
</tr>
<tr>
<td>M6</td>
<td>( B \rightarrow A ): ( R_B, f_{K_A}(A, B, R_A, R_B), f_{K_C}(A, B, R_A) )</td>
<td></td>
</tr>
<tr>
<td>M7</td>
<td>( A \rightarrow B ): ( f_{K_C}(A, B, R_B) )</td>
<td></td>
</tr>
</tbody>
</table>

If the Lin protocol is converted to use elliptic curves, there would still be one more ECC multiplication performed on both sensor nodes \( A \) and \( B \), as well as extra random number calculations (for the mapping of the key to an elliptic curve point).

TinyOS has a seven byte overhead when sending a single packet, the attributes include:

- Address is two bytes.
- Type is one byte.
- Group is one byte.
- Length is one byte.
- CRC is two bytes.
In a BSN the number of sensors is dramatically less than the number of sensors in a large scale WSN. In a BSN the address can be one byte, and the group can be removed. Also, if the message contains a MAC, then the CRC can be removed as well. So the packet overhead in a BSN is between three bytes to five bytes. Also in TinyOS the data size of the packet can be changed. To simplify the analysis an assumption is made that the data size is greater than the largest message that needs to be sent.

Table 5.9 shows the comparison of the Lin protocol, CSPK protocol, PNEKA protocol, NEKAEV protocol and the BTP Protocol. Most of the values for the Lin protocol were obtained directly from their publication [105]. The number of messages and bytes sent by each sensor node is also included. Since the proposed protocols are developed as extensions to either the Boyd protocol or the Bellare–Rogaway protocol, these two protocols were not included in the table. An assumption is made that the $\Phi$ attributes in the proposed protocols is calculated via an encryption algorithm, rather than a MAC. If the proposed protocols were using a MAC to calculate the $\Phi$ values, then it would be necessary to move the encryption values down to the MAC row. The nonces are simple counters, and the random numbers shown in this table is for the creation of any new keys. A sensor node sending data is a very expensive operation: each bit transmitted consumes the same amount of energy as thousands of CPU instructions, if the distance between the nodes increases, then so will the amount of energy required.

<table>
<thead>
<tr>
<th></th>
<th>Lin</th>
<th>BTP</th>
<th>CSPK</th>
<th>PNEKA</th>
<th>NEKAEV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>S</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>messages sent</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>modular exponentiation</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>en(de)cryption</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MACs</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>random numbers</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bytes sent</td>
<td>51</td>
<td>61</td>
<td>52</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The BTP Protocol sends the least number of messages. However, it can only be used if a $K_S$ exists to refresh the keys between $A$ and $B$. The number of encryption and decryption operations in the Lin protocol is the same or less than the number of encryption and de-
CHAPTER 5. ESTABLISHING INITIAL KEYS

cryption operations required by the proposed protocols. However, the proposed protocols do have the advantage of removing any need for block cipher encryption and decryption operations, by using simple exclusive-or with either a random nonce or a Φ value. The number of MACs calculated by the Lin Protocol for each of its sensor nodes is either worse or the same as the number of MACs calculated by the proposed protocols for each of their nodes. The proposed protocols based on the Boyd four-pass protocol and the two-pass protocol have an extra MAC calculation, since $K_{AB}$ is calculated using Equation (5.5). Finally, all the protocols, except for BTP protocol, require a good random number to be calculated to generate $K_{AB}$, $K_S$, or the ephemeral keys. Table 5.9 shows that BTP protocol is the most efficient protocol in terms of number of messages sent, whereas, the Lin protocol is the least efficient. A useful aspect in the CSPK protocol, PNEKA protocol, and NEKAEV is that, whenever a need arises, the key $K_{AB}$ can be refreshed using the BTP protocol.

This section next compares the number of cryptographic operations required when performing multiple updates of key $K_{AB}$. For simplicity an assumption is made that each operation costs the same amount. The number of modular exponentiations, encryption, decryption, MACs, and random numbers is also added. The number of updates to the key is then multiplied to the above value. In the Lin protocol there is $28m$ operations, where $m$ is the number of iterations of the protocol. The number of cryptographic operations performed by the CSPK protocol is $25m$. The PNEKA protocol, which is derived from the Boyd protocol, has a total of $2m + 14$ operations; however, one of the message exchanges does require a cryptographically strong key. The NEKAEV protocol, which is also derived from the Boyd protocol, has a total of $2m + 14$ operations; however, two message exchanges require cryptographically strong keys.

The next comparison is the number of messages sent by each protocol. The Lin protocol has a total of $7m$ messages, where $m$ is the number of iterations of the protocol. The number of messages required for the CSPK protocol is $4m$, a significant reduction if the sensors need to regularly update their keys. The number of messages for the PNEKA protocol and NEKAEV is $2m + 2$. The first iteration of the protocol is 4, as seen by the total number of messages in the PNEKA protocol and the NEKAEV. The second iteration uses the BTP protocol, which only has two messages. Once again, the existing security requirements before the running of the PNEKA protocol and NEKAEV are slightly different, in terms of the requirements of the key strength.

The number of messages sent is not a true indication of the performance of the protocol, since the number of bytes sent in each message may be different. When investigating the amount of bytes sent, the Lin protocol was found to have a total of $164m$ bytes. An assumption is made that the nonces are 1 byte, MACs are 4 bytes, location addresses are 1 byte, session keys are 8 bytes, and the exponent is 40 bytes. The number of bytes for the CSPK protocol is $127m$. The number of bytes sent by the PNEKA protocol is $2m + 108$. The number of bytes for NEKAEV is $2m + 48$. 

5.9. ANALYSIS AND IMPLEMENTATION

5.9.1 Implementation

Previous research in sensors suggested that RSA is not suitable, and elliptic curves should be used [116, 178]. We extensively examine this claim and investigate if RSA can be efficiently implemented. When developing the security mechanisms the assumption was made that an adversary is not able to obtain the SEVs. This enables the RSA Diffie–Hellman implementation to use lower prime numbers. The EKE protocol can use a 160-bit exponent, as long as the adversary does not obtain the SEV used in the protocol.

To convert the RSA implementation of a password protocol into an elliptic curve implementation, the SEV will need to be mapped to an elliptic curve point. The mapping requires a much larger key than the 160-bit exponent required for the above protocol, even if the SEV is not discovered by an adversary.

An ECC implementation for 8-bit ARM microcontrollers with low computational power and low memory resources is required. Several implementations for TinyOS [102] have been developed, such as the Malan’s implementation [116] or the Deluge implementation [59]. Most implementations of ECC are for 32-bit or 64-bit architectures, where the memory constraints found in the low-end sensors do not exist. Finding an efficient ECC implementations for 8-bit ARM microcontrollers is still an open research question.

This thesis uses existing implementations rather than creating a new implementation. The results found in this thesis may change if more efficient implementations are discovered and used.

The modified PPK protocol and the CSPK protocol were implemented in TinyOS [102] and simulation was run using TOSSIM. Malan et al. [116] implementation of elliptic curve for sensor networks was used. The elliptic curve $y^2 + xy = x^3 + x^2 + 1$ and reduced polynomial $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$ is used. The simulation time for the two party was 13.38 seconds. The simulation time for the three party protocol was 17.92 seconds. The simulation results for the two protocols are shown in Table 5.10, where energy comparisons are made between the sensor nodes and the protocols. All the values are given in microjoules.

<table>
<thead>
<tr>
<th>ID</th>
<th>ECC radio</th>
<th>ECC CPU</th>
<th>ECC total</th>
<th>ECC &amp; SKIPJACK radio</th>
<th>ECC &amp; SKIPJACK CPU</th>
<th>ECC &amp; SKIPJACK total</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>78.56</td>
<td>31.54</td>
<td>110.10</td>
<td>131.68</td>
<td>52.89</td>
<td>184.57</td>
</tr>
<tr>
<td>B</td>
<td>80.10</td>
<td>32.16</td>
<td>112.26</td>
<td>117.72</td>
<td>47.27</td>
<td>164.99</td>
</tr>
</tbody>
</table>

A memory distribution comparison of the implementation of the CSPK protocol is shown in Table 5.11. The .bss and .data segments use SRAM, and the .text segment uses...
ROM. When elliptic curve cryptography is used the size of the application for both ROM and RAM has doubled in size.

Table 5.11: Memory Overhead in Bytes on MICA2 Platform

<table>
<thead>
<tr>
<th>Memory</th>
<th>ECC</th>
<th>No ECC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
<td>40 130</td>
<td>25 470</td>
</tr>
<tr>
<td>RAM</td>
<td>1478</td>
<td>761</td>
</tr>
<tr>
<td>.data</td>
<td>264</td>
<td>264</td>
</tr>
<tr>
<td>.bss</td>
<td>1214</td>
<td>497</td>
</tr>
<tr>
<td>.text</td>
<td>39 866</td>
<td>25 206</td>
</tr>
</tbody>
</table>

The major advantage that PNEKA protocol and NEKAEV, which are based on the Boyd protocol, have over the CSPK protocol, is that the first two protocols have an extra key. This key can be solely used to establish new keys between A and B. To update a key when using the CSPK protocol, a two party protocol, such as the SEVN protocol or the modified Anderson–Lomas protocol, would be required.

An implementation and comparison was completed on the proposed group key agreement protocol on a Crossbow mica2 MPR2600 mote. The protocols were developed using TinyOS 2.x. The RSA code was based from the Deluge System, and a 160 bit exponent is used as required by most RSA based password protocols. Previous studies assumed that a larger key size is needed when using RSA [175]. However, password protocols have different characteristics where the assumption is not valid. The experiment had one sensor attached to a workstation while the other sensors were placed stand-alone. After the running of the protocols, the elapsed time was then sent via the serial connection, to a PC running a Linux® distribution where there is a Java® application reading the TinyOS packet from the serial port, and report that data to the user.

The configuration that was used is shown in Figure 5.3. One of the sensors is attached to the computer with a USB cable. The computer registers the connection as a serial port. The communication between the computer and the sensor is achieved through the serial port.

Table 5.12 shows the ratio of the application speed for each of the algorithms compared to exclusive-or algorithm. When the algorithm ran on the mica2 mote, time taken to run 2000 iterations was approximately one millisecond. In the ATEMU simulator it took approximately 7000 instructions.

Little difference was found between the simulation results and the amount of time an operation takes when put on the mica2 mote. The most notable difference in results was for the ECC algorithm where there was a two percent difference. Both the time and number of instructions suggest that for the same size key the RSA algorithm is significantly better than the ECC algorithm.
5.9. ANALYSIS AND IMPLEMENTATION

Figure 5.3: Reading from the Sensor using the Serial Port

Table 5.12: Comparison of Application Speed: As Ratio to Exclusive-Or

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mica2</th>
<th>ATEMU</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC5</td>
<td>453</td>
<td>456</td>
</tr>
<tr>
<td>SKIPJACK</td>
<td>739</td>
<td>741</td>
</tr>
<tr>
<td>HMAC–MD5</td>
<td>18 400</td>
<td>18 500</td>
</tr>
<tr>
<td>RSA</td>
<td>41 600</td>
<td>41 900</td>
</tr>
<tr>
<td>SQRT</td>
<td>87 800</td>
<td>88 400</td>
</tr>
<tr>
<td>ECC</td>
<td>4 820 000</td>
<td>4 920 000</td>
</tr>
</tbody>
</table>
5.9.2 Memory Size

The size of the application both in terms of number of lines of code and the size in bytes is important when choosing an algorithm. Table 5.13 lists the number of lines of code and the size in bytes of the application that was used to run the original time and instruction measurements.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Code Lines</th>
<th>Size (bytes)</th>
<th>Stack (bytes)</th>
<th>RAM (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR</td>
<td>80</td>
<td>5600</td>
<td>158</td>
<td>432</td>
</tr>
<tr>
<td>RC5</td>
<td>506</td>
<td>6776</td>
<td>172</td>
<td>466</td>
</tr>
<tr>
<td>SKIPJACK</td>
<td>697</td>
<td>8138</td>
<td>190</td>
<td>496</td>
</tr>
<tr>
<td>RSA</td>
<td>1456</td>
<td>7062</td>
<td>213</td>
<td>624</td>
</tr>
<tr>
<td>SQRT</td>
<td>3366</td>
<td>7662</td>
<td>230</td>
<td>748</td>
</tr>
<tr>
<td>ECC</td>
<td>5038</td>
<td>14020</td>
<td>760</td>
<td>2066</td>
</tr>
</tbody>
</table>

The Code Lines indicates lines of code and thus the complexity of the code for a developer to implement the application. The Size (bytes) indicates the size in bytes of the application. The Stack (bytes) indicates the maximum size of the stack for the application. The RAM (bytes) is the maximum amount of RAM the application will need. The figures are obtained from the stack analysis tool found in TinyOS.

The RC5 application took considerably more time to implement than the XOR application. A RC5 implementation was found for TinyOS 1.x in the TinySEC library [95]; however, the code needed to be ported to TinyOS 2.x. A port of the code was completed for the new platform.

The SKIPJACK application had similar problems as the RC5 application. Where there was an implementation for TinyOS 1.x in the TinySEC library but there was not one for TinyOS 2.x. Once again, significant effort was put on porting the code to the platform.

The RSA application also had similar problems as the RC5 and SKIPJACK implementations. Code was found in the Deluge System [58]; however, the RSA code was based off TinyOS 1.x. Effort was required to port this code to TinyOS 2.x. A 160 bit exponent was used as required by the EKE protocol.

The SQRT application had the most difficulties since that was implemented from pseudo-code rather than porting any code. Newton’s Method [131] was used for finding square roots to implement the SQRT application.

The ECC application also had similar problems to the RSA, RC5 and SKIPJACK implementations. A port of an ECC library [107] developed for TinyOS 1.x was completed for TinyOS 2.x. The ECC application used a 160 bit elliptic curve points, since password protocols that could be converted to use ECC require stronger keys.
5.9. ANALYSIS AND IMPLEMENTATION

The XOR application is the quickest by several orders of magnitude compared to the other cryptographic primitives. But the size of the application is smaller, and the number of lines is less than the other applications. The XOR application is the quickest, whereas the ECC application is the slowest. This verifies the previous sections into the differences in speed for password protocols of RSA and ECC implementations in TinyOS simulators. The HMAC–MD5 application is the largest; however the application was a straight port from the RFCs, where the code was not intended for sensors.

5.9.3 Protocol Times

Even though exclusive-or and block cipher symmetric cryptography is suitable in an RSA environment when using the EKE protocol, it is not suitable when converting to elliptic curves [33]. The EKE (RSA) protocol is compared with a ECC based password protocol, called PPK [34].

The time taken for the proposed RSA based protocol was measured when executed on the mica2 mote system, and using 160 bit exponents. The PPK password protocol using an ECC implementation require a key size of 160 bits since in RSA mode 1024 bit exponents are needed [33]. When moving to the ECC protocols, more secure keys are required. The total time taken of ECC implementation was also measured, the measurement also includes an execution of the square root function. There is a significant extra overhead in a ECC implementation over the RSA implementation, as shown in Table 5.14.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>bytes</th>
<th>packets</th>
<th>time (msecs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR</td>
<td>10</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>EKE</td>
<td>45</td>
<td>4</td>
<td>102</td>
</tr>
<tr>
<td>PPK</td>
<td>69</td>
<td>6</td>
<td>4910</td>
</tr>
</tbody>
</table>

It was found that the performance costs of the cryptography algorithms was insignificant compared to the costs of communication. Where the communication can be measured in the milliseconds the computation costs had to be measured in the microseconds range. On average each message sent took 10 ms to be received and parsed by the opposite end. When the protocol was run over four nodes, the total time taken was found to be 97 ms. This research has yet to try this protocol with a larger number of sensors.

The memory used by the protocol was also investigated. An application without RSA was developed. The application only sent and received messages. The parsing and calculations of the group key was added into the code. It was found that the application without RSA had a maximum RAM size of 411 bytes. By adding the RSA algorithm into the application the RAM size grew to 624 bytes.
5.10 Conclusions

This chapter and its contributions have mainly tested the Hypothesis 2, which also helps to answer the Research Question 2. A new key establishment protocols has been proposed for both two party and three party scenarios. The proposed protocols extended basic traits from a two party password-authenticated key exchange protocol and a symmetric key server-based key establishment protocol. The proposed protocols have the advantage that they do not require traditional encryption to transport the new session key. This chapter showed how the three party protocol can be used for cases where two sensors read different types of physiological data. A comparison study was performed, revealing the requirements of different cryptographic operations. The NEKAEV protocol does not require any modular exponentiation. The BTP protocol is another example, which does not need any modular exponents, but does require an existing key \( K_S \). If a key establishment protocol creates a key \( K_S \), then the BTP protocol is an efficient mechanism to create new session keys between two nodes.

In this chapter, a secure group key establishment mechanism was developed and proposed for a body area network. It allows multiple devices to agree on a shared group key, in an authenticated manner, without the need for any form of initialisation or \textit{a priori} knowledge. This chapter showed that designing a secure group key agreement protocol is not a trivial task, and discussed different types of attacks. The proposed group protocol uses SEVs to achieve authentication. The protocols were implemented in TinyOS and run on mica2 motes. The time elapsed and memory usage showed that the protocol meets the general BSN requirements. This chapter showed that a group key establishment protocol can be efficiently run in a sensor network with the RSA algorithm.

This chapter implemented salient features of the protocols and compared the energy consumption of the nodes. It was found that when using ECC the communication cost was nearly doubled; the RSA protocols that could be converted to use ECC required large prime numbers. Also, when mapping data to elliptic curve points caused the protocol to become non-deterministic in energy costs. There is a security gain in using password protocols that can be converted to ECC; however, the security gains are not significant if the exponents in RSA are chosen wisely. The extra requirements needed to enable ECC caused these protocols to be prohibitive. The impact on memory by adding elliptic curves to a sensor application was analysed, revealing that there is additional costs associated with an ECC solution over an RSA solution.

The analysis of the protocols designed for a BSN did not validate that the assumptions used for the proposed protocols are valid in a mobile health care system. Also, another problem with the analysis is that it did not include an analysis of any physiological data. Research has shown that over time physiological data can create cryptographically secure keys. Research showing the entropy of physiological data over a short period of time is
required. Another hindrance to the system is that most of the research use ECG; however, ECG is cumbersome since it requires two points of contact. In the following chapter, a new method will be shown to validate protocol assumptions. The next chapter will also validate that low entropy keys can be generated from actual physiological data. An implementation using a pervasive mobile phone as well as an Arduino device is shown.
This chapter will validate the security protocols described in the previous chapters. A mobile health care system is described, some of the security protocols from previous chapters are mapped into the mobile health care system, as an example of which components act as a server and which components act as a sensor. The example protocols are then described in GDM by using requirement behaviour trees. This chapter then shows how security assumptions found in each of the proposed protocols can be validated in the mobile health care system. Further validation is done on showing the actual entropy values of physiological data. A database of ECG values is used; a security analysis of the protocols is carried out, where this research has the assumption of worst case scenarios. The validation step resulted in a new family of security protocols to be developed and designed, where the sensed data has a level of uncertainty or error margin. The new family of security protocols take the level of uncertainty into account. The final validation step is an implementation of sensing the physiological data using pervasive smart phones, and low-cost Arduino devices.

6.1 Introduction

Sensors can be used to remotely monitor elderly or patients suffering from chronic diseases and allow them to have relatively independent lives. Proposed health care systems contain many different components and hence are inherently complex [99, 106]. The complexity hinders security proofs and detailed analysis, so much so that theoretically proving that
a protocol is secure within an entire system is rarely performed. Instead researchers only try to show that a sub-system is secured, for instance, communication between two body sensor nodes. Another problem is that complex systems rarely stay static; with new devices and algorithms a system may have different functionality from one day to the next. A proof that a protocol is secure for one system does not guarantee that it is secure on the other system.

In the previous chapters the components of the mobile health care system were described. In this chapter more detail about the flow of data and the computation at each component is described. Figure 6.1 demonstrates a method to model a mobile health care system proposed in this thesis. For the data flow analysis the body sensors, home sensors and mobile phone are used to gather information from the patient. The sensed data are then passed onto the home health controller. The home health controller analyser module will then analyse the data and send the analysis results to the decision module and hospital. The decision may be performed either at the home health controller decision module or at the hospital, depending on the results of the analysis. Once the decision is made it needs to be acted upon, this feedback may be a message sent to the patient that they should come in for an extra check-up or the sensors need to start measuring physiological data at higher sampling rates or that nothing needs to change.

![Home Health Care System Data Flow Diagram](image)

Figure 6.1: Home Health Care System Data Flow Diagram

A patient at home can have a number of body sensors that can communicate with home sensors, the health controller and a mobile phone. Home sensors, such as cameras, may only start recording if the body sensors detect that there may be a medical emergency, such as the patient lying horizontal in the kitchen. Surveillance software, such as S3 [73], can be used to detect if the patient is cleaning the kitchen or getting something from the ground or there is actually an emergency. If the software does detect an emergency, the hospital staff is notified, they examine the information and decide on the best course of action. The mobile phone is used to give feedback to the patient about the condition of their body, as well as the status of the sensors. The mobile phone can notify the patient of any detected emergency, allowing the patient to report back a false alarm if one has occurred. The mobile phone can be replaced with a PDA or any other hand-held
6.2 USE OF GDM IN COMPLEX SYSTEMS

Communication device.

Health information collected from sensors needs to be secured and in some countries (for example the USA) security is mandated [167]. Securing a mobile health care system becomes more challenging mainly because of the different requirements placed on various components in the system. For instance, the sensors have severe resource constraints than the constraints found in mobile phones, cameras or desktop computers. With differences in computing power, as well as in communication costs, a range of security protocols may be required to be deployed in the entire system. For instance, an efficient key establishment mechanism specifically suited for body sensors was proposed using physiological data in the previous chapter. However, the mobile health care system may send physiological data to medical staff or to an analytic engine [62]. The physiological data may also be sent to an actuator to release medicine into the body [62]. These may jeopardise the ability to use the physiological data in the key establishment protocol.

When the same physiological data is used for multiple purposes and the environment is heterogeneous and complex, it becomes important from a security or information assurance point of view to have a formal methodology to validate the system. A formal methodology is also important to insure that the information sent to medical staff and actuators to dispense medicine is accurate (secure), and the correct actions are taken. The formal methodology has a requirement that it can model the security and privacy aspects as well as the assumptions and the application correctness.

In Section 6.2 there will be a description on how Genetic Design Methodology (GDM) is currently used to model complex systems. Section 6.3 supplies a description of a mobile health care system, and some of the components that belong in it. In Section 6.4 there will be a description on key establishment protocols for a complex sensor system. The security of the key establishment protocol will be discussed and how it is dependent on some assumptions about the environment. In Section 6.5 GDM will be further described and how it can be used as a formal analysis tool to verify both the system and in particular the assumptions made by the security protocols are correct. In Section 6.7 supplies a validation that the IPIs can be used as low entropy keys. Section 6.8 concludes the chapter.

6.2 Use of GDM in Complex Systems

The common techniques used to verify that a protocol is correct do not easily scale to a complex system. Using the above techniques to validate a system with hundreds of nodes, and many different protocols, is almost unfeasible. To further complicate matters, there are inherent restrictions such as; the formal verification will need to be repeated for every minor change in the system.

When proving that a protocol is secure, the proof relies on a number of assumptions
made about the environment where the protocol is run. These may be assumptions such as secure time synchronisations between the parties, or that the physical security of the communication medium. Showing that a protocol is secure in a complex system may be considered as a two step process. The first step is proving that the protocol is secure based upon some assumptions. The second step is to show that the assumptions are valid and consistent in a complex system. The GDM has recently been used as a tool to prove and/or validate the overall correctness of complex systems.

As reported earlier, Sithirasenan et al. [160] have used GDM to check the correctness of the 802.11i wireless security protocol. The requirements of the protocol were placed into a number of Requirement Behaviour Trees. The requirements were then verified by integrating them into a single Integrated Behaviour Tree. Thereafter, the Behaviour Tree model was translated into SAL formal notations for theorem proving. This mechanism shows that both model checking and theorem proving approaches can be performed using the same analysis tool. The checks performed was mainly focused on the protocol correctness and not on the assumptions made by the protocol. This chapter will show that the GDM analytical tool can effectively perform model checking on the correctness of protocol assumptions in a complex system.

When using GDM, systems are designed out of the requirements as opposed to methods that produce designs to meet the requirements. A major advantage of GDM is that it produces graphical models that are derived and integrated from the original requirements. The models can easily be used to verify that security protocols correctly work in a complex system.

An example of a complex system is the mobile health care system. For instance, in a mobile health care system it can become difficult to track how sensed data is used at different stages in the system. When the sensed data is also used in key establishment protocols, tracking the various uses of sensed data becomes even more important. For example, some key establishment protocols require that the sensed data never to be sent in the clear or to an untrusted third party, whereas other protocols do not need such restrictions. The complex system and the protocols can be defined in requirement behaviour trees using GDM.

As described in Section 2.2.2, behaviour trees provide a direct and clearly traceable relationship between what is expressed in the natural language representation and its formal specification. Conventional software engineering method applies the underlying design strategy of constructing a design that will satisfy its set of functional requirements. Whereas, a clear advantage of the behaviour tree notation is that it allows the construction of a design out of its set of functional requirements, by integrating the behaviour trees for individual functional requirements (RBTs), one-at-a-time, into an evolving design behaviour tree (DBT).

The RBTs are integrated based on the precondition of the tree that must be satisfied.
6.3. ANALYSIS OF THE HOME HEALTH CARE SYSTEM

in order for the behaviour encapsulated in a functional requirement to be accessible or applicable or executable. If there is no matching post-condition embodied in the evolving DBT, then a defect is identified and needs to be rectified. In which case, either the requirement is invalid, or there is a missing requirement. Integrating RBTs is an important feature when showing that security requirements in the system are valid or if there is a missing security requirement. Once the RBTs are integrated using IBTs, and any missing or invalid requirements are dealt with, other models can then be generated from the evolved DBT. SAL code can be generated; allowing the creation of theorems that also checks the security requirements of the system.

Behaviour trees can in turn be used to generate SAL code [160]. A model checker can then be used to verify the SAL code and thus verify the protocol in the sensor environment. The main steps in the GDM are:

- Translation of requirements to behaviour trees.
- Integration of behaviour trees.
- Architecture transformation.
- Component behaviour projection.
- Component design.

When modelling the entire system, genetic design has significant advantages over Unified Modelling Language (UML), state charts or other methods [55]. The advantages of using GDM include the ability to allow designers to focus on the complexity and design of individual requirements while not having to simultaneously worry about the details in other requirements. The requirements can be dealt with one at a time (both for translation and integration). Another advantage is the component architecture and the component behaviour designs of the individual components are emergent properties of the design behaviour tree.

The methodology concentrates on discovery of behaviour gaps, which in turn discovers requirement and security gaps. The focus of direct translation of requirements to design makes it easier to see and find gaps either manually or using automated tools. The GDM provides an automated method of mapping changes in requirements to changes in design. The use of GDM specifically in the security field has not been explored. Further research in this field is required to check the suitability of using GDM on security protocols. In particular, this chapter will show that the GDM analytical tool can effectively perform model checking on the correctness of protocol assumptions in a complex system.

6.3 Analysis of the Home Health Care System

The mobile health care system, described in this chapter, is relatively complex. When security mechanisms are incorporated, the importance of modelling technique becomes
CHAPTER 6. VALIDATION

more apparent. This section examines in detail a complex security protocol that secures a hand-held device, such as a PDA or mobile phone with the mobile health care system.

Table 6.1 depicts an overview of four of the major components in mobile health care system: mobile phone; home sensors; home health controller; body sensors. The table shows the different types of communication protocols and technology each of the components may have, as well as the physical security of the component. The connectivity column describes when or how often the component is able to obtain patient information and the replacements column indicates how often the patient will replace that particular component. The operations of each component are described below.

Table 6.1: Components in the Health Care System

<table>
<thead>
<tr>
<th>Component</th>
<th>Communication</th>
<th>Physical Security</th>
<th>Connectivity</th>
<th>Replacements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body Sensors Implant</td>
<td>802.15.4</td>
<td>Very High</td>
<td>Always</td>
<td>Very seldom</td>
</tr>
<tr>
<td>Body Sensors Strapped</td>
<td>802.15.4</td>
<td>Medium</td>
<td>Always</td>
<td>Frequently</td>
</tr>
<tr>
<td>Mobile Phone</td>
<td>802.11, 802.15.4</td>
<td>Low</td>
<td>When turned on &amp; close to patient</td>
<td>Frequently</td>
</tr>
<tr>
<td>Home Sensors</td>
<td>802.11 and/or 802.15.4</td>
<td>High</td>
<td>When patient is at home</td>
<td>Frequently</td>
</tr>
<tr>
<td>Home Health Controller</td>
<td>802.11, 802.15.4</td>
<td>High</td>
<td>When turned on</td>
<td>Seldom</td>
</tr>
</tbody>
</table>

6.3.1 Body Sensors

Body sensors measure the vital physiological signs of the patient. Body sensors use a low powered communication medium such as 802.15.4. The implanted sensors' physical security is very high and they should be rarely replaced. Sensors that are strapped onto the patient are less secure and can be replaced more frequently. For the remainder of this chapter an assumption is made that the patient has at least one implanted sensor. The implanted sensor can be used to safely store security keys within the home health care system.

The sensors will need to send the information securely. This produces an inherit requirement of establishing session keys for the sensors. After sensors gather the data they will need to send that data to a central computer via a mobile phone or the nearest component within the system, as described in Figure 6.1.
6.3. ANALYSIS OF THE HOME HEALTH CARE SYSTEM

6.3.2 Mobile Phone

The mobile phone is a mobile gateway between the body sensors and the rest of the network. It can be incorporated to use many different communication technologies. It can display the status of all sensors and the vital signs for the patient. The display will facilitate the patient to feel in control of the entire system.

Its mobility causes it to be easily lost or misplaced. Hence the physical security of this device is low. Therefore, an important limitation placed on this device is that any cryptographic session keys must not be stored in stateful memory. Therefore, the mobile phone will need to be able to quickly establish secure session keys with the body and home sensors when it is turned on.

For privacy reasons when the patient goes outside their home, the phone should be able to send data back to the home health controller. The home health controller can decide, based on a set of pre-determined rules, the salient information that needs to be sent to the hospital staff.

The mobile phone also has a sensor attached to it. Hence if the patient picks up the phone, the phone will be able to read the physiological data from the body. This is a convenient method to have redundant sensors on the body to accommodate the case if one of the sensors becomes faulty. By having a sensor on the phone, the system can also quickly detect any faulty sensors.

6.3.3 Home Sensors

The home sensors can augment the body sensors and supply more information, such as temperature and movement of patients. The home sensors may not always be switched on, and may only be turned on if the mobile health care system determines that there may be an emergency and requires more information. This saves power and also enhances the privacy of the patient. Especially if some of the home sensors are cameras.

The home sensors need to establish session keys with the mobile phone and body sensors. The mobile phone can be used by the patient to know the status of the other sensors in the system. The body sensors should directly be able to communicate with other components within the system, allowing the patient to freely move within their home without needing to always carry their mobile phone.

Since the home sensors are located inside a building they are physically secure and have a lower likely-hood of being stolen. Home sensors will only be replaced if they are found to be faulty. This thesis has an assumption that the security between the home sensors and the health controller is achieved by other known protocols, such as defined in the IEEE standards for 802.11 [82]. This chapter focuses mainly on securely establishing keys with body sensors.
6.3.4 Health Controller

The health controller coordinates the entire mobile health care system. It contains heuristics to determine if the patient is in any danger. If the health controller determines that the patient is in some danger, it may then power on some of the home sensors to gather more data. It may also notify the patient or the hospital depending on the type of danger. The home health controller is a key component for the privacy of the patient. The patient is more likely to have cameras installed in their home with the assurance that they will only be turned on in case of an emergency.

The health controller has an interface allowing the patient to enter extra data, such as, going for a jog, cleaning, etc. This enables the health controller to obtain an informed judgement on whether the patient should go to the hospital for an early check-up, or if the patient is doing some exercise.

The health controller is the central hub within the home; hence it will need to have established session keys between all of the other components. The health controller also has a built-in body sensor similar to the mobile phone. This allows the patient to place their hand on the sensor to confirm that the readings from the body sensors are accurate or whether any of the sensors is faulty and needs to be replaced.

6.3.5 Other Components

Other entities within the system include the hospital and the patient, as shown in Figure 6.1. The patient originates the physiological data, which the body sensors can capture. The patient can also input information to the system, such as exercising or eating, via a mobile phone or PDA.

The hospital will be supplied salient information about the patient. However, the patient's privacy will not be compromised by supplying non-vital data (such as, camera footage), unless there is a justified medical reason for it. The hospital will be notified of any irregularities and hence the hospital can inform the patient to have an additional appointment with a doctor, etc.

6.4 Proposed Mechanisms for Key Establishment

There are several scenarios where the body sensors and the mobile phone do not have any keys established with the remainder of the mobile health care system. This may be because the mobile phone has recently been turned on, and no keys are stored in the mobile phone's permanent memory. Hence it will need to establish keys with the rest of the system a fresh. The body sensors may have expired keys, if the patient was away from their home for an extended period of time. For operational efficiency reasons, keys
in body sensors are small; hence the keys have a small life-time and need to be updated more frequently.

There are four major steps when first establishing keys, the steps are as follows:

- **Initial Setup**: This step describes how the system is initially setup.
- **Home Health Controller and Body Sensor**: This step describes how the Home Health Controller establishes a key with the body sensor.
- **Body Sensors and Home Sensors**: This step describes how the body sensors establish keys with the home sensors.
- **Mobile Phone**: This step describes how the mobile phone establishes keys with the other components.

Each step is described in detail in the following sections.

### 6.4.1 Initial Setup

In this chapter, an assumption is made that the session keys between the Home Health Controller and the Home Sensors have already been established [82]. The assumption is that existing industry standards, such as 802.11, was used to set the system up. The standard 802.11 is well understood and found in many higher resource environments, such as, mobile phones, cameras, laptops, and personal computers.

### 6.4.2 Home Health Controller and Body Sensor

The next step is to establish a session key between the Home Health Controller and a body sensor. An implanted body sensor is desirable since they have the highest level of physical security. That body sensor can be used in the future to hold session keys for most of the other components in the health care system.

Ease of use is an important requirement for the system. A complex system where a user needs to install security certificates is unfeasible. Setting passwords on tens to hundreds of devices is technically difficult due to lack of a user interface such as keyboards or monitors on many of the sensors. Forcing a patient to remember a new password or PIN will be a deterrent on the take up of a mobile health care system. It is envisioned that the elderly could benefit from a mobile health care system. However, they would have the greatest difficulty in managing and handling complex technology. As technology improves new devices will be added to an already complex system. It is unfeasible to have users learn new technology when changing components in an ever changing environment. The security setup should be simple so any new devices can be easily added.

An example of a simple method to establish a key between two devices is for the patient to place their palm on a sensor (such as an ECG reader) built into or attached to the Home
Health Controller. If an ECG reader is used then authentication between the Home Health Controller and the patient can also be performed [40, 164].

The previous chapter has supplied an overview of key establishment within a BSN and have shown different methods to establish keys between sensors and components that obtain the same SEV:

- Use an SEV as a one-time pad.
- Use an RSA-based Diffie–Hellman password protocol, where the password is the SEV.
- Use an ECC-based Diffie–Hellman password protocol, where the password is the SEV.

The ECC-based password protocols have major efficiency problems, which are shown in the implementation. Hence in this section the first two approaches are examined in more detail.

The simplest method is to use the SEV as a one-time pad. Venkatasubramanian et al. [173] used a single message to send a new key to the neighbouring sensor node, by using the SEV as a one-time pad, as shown in Protocol 6.1. The home health controller, $C_h$, initiates the protocol by sending a message to the implanted sensor, $S_b$. The new session key is sent encrypted by the SEV, $K_{C_hS_b} \oplus \Upsilon$; where $K_{C_hS_b}$ is the newly created session key and $\Upsilon$ is the SEV read by each of the sensors. Venkatasubramanian et al. noted that finding additional cryptographically sound physiological values is still an open research problem. Another problem is that all the protocols developed with physiological values require all the sensor nodes to be able to measure the same phenomena. Only cryptographically strong physiological values, such as IPI (Inter-Pulse Interval) and HRV (Heart Rate Variance), can be used. Also, modern wireless technology (ultra wideband — UWB, radar [162]) may be used to remotely capture the heart rate and could cause security risks when using IPI and HRV to only secure the communication. Other cryptographically weaker physiological values, such as blood pressure, and iron count, are less susceptible to those remote attacks.

**Protocol 6.1** Venkatasubramanian BSN protocol for mobile health care

$$C_h \rightarrow S_b : N_A, [N_A]_{K_{C_hS_b}}, K_{C_hS_b} \oplus \Upsilon$$

The new key $K_{C_hS_b}$ is encrypted with the physiological value $\Upsilon$, which is only known to sensors on a particular person. Sensor node $B$ validates that $K_{C_hS_b}$ is correct by verifying the MAC of $N_A$.

One of the major limitations of this protocol is the length of time it will take to generate a sufficiently random physiological value $\Upsilon$. For example, a sufficiently random value based on IPI will take approximately 30 seconds [172]. Another limitation is that the Venkatasubramanian et al. protocol has a requirement that the sensed data should never
be sent in the clear or to an untrusted third party.

A more complex method is to use an RSA-based password protocol as described in the previous chapter. It was shown how the EKE protocol can handle small entropy secrets, so that off-line and on-line dictionary attacks are unfeasible for an adversary. The smaller entropy secrets do not require a long time to generate. If an ECG/PPG peak is obtained every 300–500 ms, a secret can be generated in less than a second. Another useful feature is that even if the SEV is compromised or available freely after the running of the key establishment protocol, the new session key will remain secure and safe. Several other RSA password protocols were also developed, in this chapter the EKE password protocol will be used.

The EKE protocol is chosen because other variants of password protocols require exponents of size 1024 bits. The EKE protocol is diagrammatically shown in Protocol 6.2, where the home health controller, $C_h$, initiates the key establishment protocol with the implanted sensor, $S_b$. A drawback of the EKE protocol is that it cannot use ECC.

### Protocol 6.2 Modified Diffie–Hellman-based EKE protocol for mobile health care

<table>
<thead>
<tr>
<th>Shared Information: Generator $g$ of $G$ where $p - 1 = qr$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_h$</td>
</tr>
<tr>
<td>$S_b$</td>
</tr>
<tr>
<td>$r_A \in \mathbb{Z}_p$</td>
</tr>
<tr>
<td>$t_A = g^{r_A}$</td>
</tr>
<tr>
<td>$K_{C_hS_b} = t_B^{r_A}$</td>
</tr>
<tr>
<td>$K_{C_hS_b} = t_B^{r_A}$</td>
</tr>
<tr>
<td>$K_{C_hS_b} = t_B^{r_A}$</td>
</tr>
<tr>
<td>$K_{C_hS_b} = t_B^{r_A}$</td>
</tr>
<tr>
<td>Verify $n_A$</td>
</tr>
<tr>
<td>Verify $n_B$</td>
</tr>
</tbody>
</table>

The EKE protocol contains four messages. The home health controller $C_h$ sends the first message to the implanted sensor $S_b$, the message contains the location specified by $A$ (the location value is in the clear), and the first part of Diffie–Hellman, $t_A$, is encrypted by the weak key $\Psi_1$. After the first message is sent, the implanted sensor $S_b$ will calculate the second part of the Diffie–Hellman scheme and hence be able to calculate the session key $K_{C_hS_b}$. The implanted sensor $S_b$ then sends the second part of the Diffie–Hellman scheme encrypted by the weak key $\Psi_2$ to the home health controller $C_h$. The nonce $n_B$ is also sent, encrypted by the session key $K_{C_hS_b}$. The last two messages authenticate both $C_h$ and $S_b$, as well as confirming that they have the session key $K_{C_hS_b}$. The encryption of $t_A$, $t_B$, $n_A$, and $n_B$ can be implemented with an XOR, as originally described by Bellovin [22].

Depending on which environmental value is measured, and how long the protocol will run, different SEVs may be used for the request and response. However, if the SEV
stays constant throughout the running of the protocol, then both \( \Psi_1 \) and \( \Psi_2 \) will be the same. The EKE protocol is designed for a constant password throughout the running of the protocol, so similar or same data for both \( \Psi_1 \) and \( \Psi_2 \) will not adversely affect the protocol.

The EKE protocol was originally designed to handle small entropy secrets, so that off-line and on-line dictionary attacks are unfeasible for an adversary. Another useful feature is that even if the secrets \( \Psi_1 \) or \( \Psi_2 \) are compromised or available freely after the running of the key establishment protocol, the session key \( K_{CS_b} \) will remain secure and safe.

Both nonces \( n_A \) and \( n_B \) are cryptographically strong random numbers, allowing the XOR function to be used for encryption. If any nonce was not cryptographically strong, then either \( n_A \oplus K_{CS_b} \) or \( n_B \oplus K_{CS_b} \) operation would allow an adversary to significantly reduce the number of valid \( K_{CS_b} \) values. A characteristic of the EKE protocol is that the nonces are never sent out in the clear, since the nonces are used to encrypt the new key \( K_{CS_b} \). The EKE protocol has the requirement that the sensed data should not be sent in the clear or to an untrusted third party, while the protocol has not completed. However, once the protocol is completed the sensed data that the protocol used can be made available.

The EKE protocol was proven to be secure [19]. However, one of the assumptions was that the shared secret is only known by the parties that wish to establish a key, and is never sent out in the clear before or during the key establishment phase.

### 6.4.3 Body Sensors and Home Sensors

The implanted body sensor can now obtain all the session keys via the Home Health Controller. Since the Home Health Controller has a secure connection between the home sensors and the implanted body sensor, it can be used as a trusted third party. Chapter 4 supplies a survey and proposed trusted third party protocols for multi-tiered sensor networks.

In this case the proposed trusted third party protocol will be used, as shown in Protocol 6.3, since it was shown to have some advantages over other protocols for the sensor environment.

**Protocol 6.3** NEKA protocol for mobile health care

\[
\begin{align*}
m &= A, B, K_C \\
\Theta_A &= [m]_{K_{Ac}}, \Phi_A = [[\Theta_A]]_{K_{Ac}} \\
\Theta_B &= [m]_{K_{Bc}}, \Phi_B = [[\Theta_B]]_{K_{Bc}} \\
M_1 &\quad S_A \rightarrow C_h : \quad A, B, N_A \\
M_2 &\quad C_h \rightarrow S_B : \quad A, N_A, \Theta_B, \Phi_B \oplus K_C \\
M_2' &\quad C_h \rightarrow S_A : \quad \Theta_A, \Phi_A \oplus K_C \\
M_3 &\quad S_B \rightarrow S_A : \quad [N_A]_{K_{ab}}, N_B \\
M_4 &\quad S_A \rightarrow S_B : \quad [N_B]_{K_{ab}}
\end{align*}
\]

There are only two messages that contain the \( \Phi \) variable; the message \( M_2 \) — server \( C_h \)
sending to node $S_B$, and the message $M2'$ — server $C_h$ sending to node $S_A$. Neither $S_A$ nor $S_B$ ever sends out $\Phi$. A possible attack on the proposed protocol is for an adversary to try and obtain the $\Phi$ variable value by interrogating $C_h$. If an adversary pretends to be $S_A$, it does not matter what locations and nonce gets passed to $C_h$, because $C_h$ should produce a new $K_C$, and therefore a new $\Phi$ and a new $\Phi \oplus K_C$.

As shown in Equation (6.1), if two exclusive-ors produce the same value, and the keys are different, then the $\Phi$ variable will have to be different.

$$\Phi \oplus K_C = \Phi' \oplus K'_C$$  \hspace{1cm} (6.1)

If the $\Phi$ variable is the same, as shown in Equation (6.2), then no extra information about $K_C$ can be obtained, as long as a strong MAC is used. It is assumed that the adversary does not know the long term key $K_{AC}$.

$$[A, B, K_C]_{K_{AC}} = [A, B, K'_C]_{K_{AC}}$$  \hspace{1cm} (6.2)

Since $S_B$ does not initiate the protocol, it has no input into the creation of $\Phi$. So an adversary who pretended to be $S_B$ has less input in the value of $\Phi$ than if they pretended to be $S_A$. The integrity of the key is also assured since key modification requires simultaneous modification of $\Theta$ as well as $\Phi \oplus K_C$.

The key $K_C$ can be used in the future to create or renew a session key between $S_A$ and $S_B$. However, that relies on the assumption that the key $K_C$ has not been compromised. Since $K_C$ is never used as a session key or used to encrypt any plaintext, the keys $K_{AC}$ and $K_{BC}$ should be compromised before $K_C$. The sensor nodes should regularly refresh $K_{AC}$, $K_{BC}$ and $K_C$ with the base station. A variant of the above protocol was proven to be secure [32]. One of the assumptions was that the long-termed key is not compromised. In a complex system it is important to know which components have high physical security and which devices can be easily obtained by an adversary.

### 6.4.4 Mobile Phone

Establishing keys between the mobile phone and body and home sensors is the last step. The mobile phone can have a sensor attached via a USB connection or have it built-in. This allows the mobile phone to establish a key with the implanted sensor in the same way that the Home Health Controller established a key with the implanted sensor.

After the key is established with the mobile phone, the implanted sensor can be used as a trusted third party to establish a key between the mobile phone and the Home Health Controller. The mobile phone can then use the Home Health Controller to establish keys between it and all the home sensors.
6.5 Modelling of the Home Health Care System using GDM

As described previously, the protocols by themselves have been proven to be secure. However, there has been no check to validate that the protocols are secure within a complex mobile health care system. The validation of the security protocol in the mobile health care system is performed by using GDM. The modelling was completed after several stages. The initial stage placed the Venkatasubramanian et al. protocol into a behaviour tree.

From the properties of the key establishment the Requirement Behaviour Trees (RBTs) are developed. While developing the RBTs, it was found that the previous definitions and properties of the protocols did not have a consistent method to define the need for the sensor to sense the physiological data. The RBT is designed for, and has built-in syntax for, external events, so this requirement was easily added to the proposed RBTs. The feature for quickly adding external events makes RBTs suitable for modelling and analysis of a sensor environment.

There are three major components in the Behaviour Tree: $C_b$; Body; $S_b$. Two requirements were put into the behaviour tree, but space restrictions limited the display of the home sensors in this chapter. While the first requirement was specified there was a missing section which was not specified in the original protocol, and that was to remove the physiological value $\Upsilon$ that was used to encrypt the new session key. The removal of the physiological value was placed that at the end of the requirement $R_1$. After the value $\Upsilon$ is destroyed the requirement $R_2$ is then attached onto the behaviour tree in Figure 6.2.

The Venkatasubramanian et al. protocol properties require that the physiological value $\Upsilon$ needs to be cryptographically strong, and the physiological value $\Upsilon$ that was used during key establishment should never be sent to a third party.

In this case another requirement is added as shown in Figure 6.3, that if the mobile phone obtains the physiological value, then it can calculate the new session key between the implanted sensor and the Home Health Controller. The same can also be done for all of the other components within the household.

There is no integration point between the first behaviour tree and the second behaviour tree. So the system is secure. A new requirement is then added, as shown in Figure 6.4. In this requirement the mobile phone has the ability to read the physiological value from the body.

After adding this requirement into the proposed system, an integration point does exist. In Figure 6.2 a $*$ is placed next to the behaviour that will be integrated with the same behaviour found in Figure 6.4. Then the requirement can be integrated, shown in Figure 6.3. The integration point will be denoted with the symbol, $\odot$. With the behaviour tree, shown in Figure 6.3, integrated into the proposed system, the system can be shown to be insecure.
Figure 6.2: Behaviour Tree Representation of the Health Care System

Figure 6.3: Intruder Represented as a Behaviour Tree
As the proposed system grows and new requirements for the system are found, past assumptions may prove to have become wrong, as shown in this case. Behaviour trees are a good tool to keep track of past assumptions, confirming that any new requirements placed on the system does not cause the system to become insecure.

The next step is to find a protocol where there are not as many restrictions. The EKE protocol is a good candidate and has the following properties:

- Sensor nodes only possess a secret of small entropy.
- Off-line dictionary attacks are not feasible.
- On-line dictionary attacks are not feasible.
- The key must have forward secrecy.

The protocol is modelled into a behaviour tree, as shown in Figure 6.5. Also, the behaviour tree (showing an intruder), shown in Figure 6.3, is no longer a valid possibility. If any component obtains the SEV they will not be able to calculate the key using the new protocol. There still exists an integration point between the mobile phone requirement, $R_4$ and the implanted sensor key establishment requirement, new $R_1$.

By using behaviour trees, all of the possible inputs and outputs that a sensor can obtain were quickly found, either through wireless communication or through their sensing devices. This also helps to verify that each component that are been developed has the needed features to run in the proposed environment. When there are a large number of sensors, this requirement becomes difficult to track. The following has been shown by first generating SAL code and then creating theorems. This section has shown that instead of generating SAL code, validation of the security protocols can be accomplished while integrating the Behaviour Trees.

This section has shown that protocol assumptions about an environment can be validated using GDM. A protocol assumption was defined as an RBT mimicking an intruder. If the intruder RBT is able to be integrated with the final IBT there is a security flaw with the system. One solution to the problem is to remove the requirement that allowed the intruder RBT to be attached to the final IBT. If that is not feasible, another solution is specifying a different security protocol with different protocol assumptions. The second
solution is used to show that a security protocol can be used even if the initial secret becomes known to an intruder.

### 6.6 Specification of SAL

Before the requirements can be tested on the key establishment protocol, the network first needs to be specified as well as the body into SAL code. To specify the network in SAL, versions of SAL libraries [135] were utilised. However, no existing SAL libraries were found to specify obtaining SEVs from the body. The body within SAL is defined as having two operations: getSEV; changeSEV. Sensors can obtain an SEV by calling getSEV and afterwards a changeSEV can be called to create a new SEV.

The SAL code is then generated from the RBTs. The first SAL code generated is for the

![Updated Behaviour Tree of a Home Health Care System](image-url)
CHAPTER 6. VALIDATION

Venkatasubramanian and Gupta protocol. Due to limitations in the SAL generation, the SAL code was modified to read the physiological data from the body SAL code. There is a requirement R2 where a sensor sends physiological data to an external third party system. It needs to be shown that requirement R2 will break requirement R1, since for the protocol to be secure it needs to ensure that the sensed data is never sent in the clear. The following theorem is used to verify that no other sensor reads the same sensed data as the pair that is establishing the new session key.

\[
\text{prop\_no\_delay: THEOREM system } \vdash \\
G(\neg((\forall (x, y : \text{principals}): \\
(\text{buffer}.1.1 = \text{buffer}.2)))
\]

SAL code was also generated for the EKE protocol. The SAL code is modified to read the physiological data from the body SAL code. There is a requirement R2 where a sensor sends physiological data to an external third party system. It needs to be shown that the requirement R2 will not break the requirement R1, since a delay was placed into the sensors in requirement R1, where the sensor will wait 30 seconds before sending out the physiological data. It should be noted that the Venkatasubramanian and Gupta protocol still is broken if the physiological data is sent out with a delay. The following theorem is used to verify that another sensor delays its send when reading the same sensed data as the pair that is establishing the new session key.

\[
\text{prop\_delay: THEOREM system } \vdash \\
G(\neg((\forall (x, y : \text{principals}): \\
(\text{buffer}.1.1 = \text{buffer}.2 \text{ AND} \\
\text{delayed}.2 = \text{true})))
\]

6.7 Validating Inter-Pulse Intervals

Another important requirement is that sensors (especially sensors that are worn) on the body may be added or removed from the body sensor network. Figure 6.6 gives a diagrammatic representation of a patient with body sensors communicating with a smart phone. The patient can carry the smart phone allowing the body sensors to be continually connected to the Internet. In the case where this system is part of a E-health system, the mobile phone will communicate with a central system. Depending on the health risks and privacy concerns of the patient, all of the information may not be transmitted to the central E-health servers.

The cryptographically secure key between the body sensor and the smart phone should be easily generated, without any intervention from the patient.
Health sensors can use Inter-Pulse Interval or Heart Rate Variance as good sources for cryptographically random numbers and the physiological values can be used as a one-time pad.

Venkatasubramanian and Gupta noted that finding additional cryptographically sound physiological values is still an open research problem. Only cryptographically strong physiological values, such as IPI and HRV, can be used. However, the method used to generate a strong cryptographic key takes approximately one minute, since 67 quantised IPI values are required. Another problem was that physiological signals measured from different areas of the body have similar trends but not necessarily the same values.

### 6.7.1 A Key Establishment Protocol using Physiological Values

As explained previously in this thesis the SEV is used to refer to sensed data that can be obtained by sensors from their environment. The sensed data is usually hard to obtain through other means. An example of SEVs can be physiological data found in the human body. The physiological data can be measured by using body sensors. A major advantage of
using SEVs is that they can be used for authentication in body sensors. A sensor on a body will be measuring physiological data that is likely to be different from that is measured on another body at any given time.

This research showed that PINs and passwords may not be feasibly in sensors, but showed that password protocols can be used. Passwords have low randomness, and therefore have similar characteristics to many of the SEVs found in the body. A four digit PIN contains less than 14 bits of randomness and can be used in a password protocol. A typical password length of eight characters has less than 48 bit of randomness, upper and lower case letters as well as the digits 0 to 9 are randomly chosen. Password protocols have the special property of allowing secrets with small entropy to be used for key establishment [33]. Password protocols are designed so that both off-line and on-line attacks are not feasible. A feature or by-product of most password protocols is that if the SEV is compromised, after the execution of the protocol, then any session key that was created will not be compromised.

Key sizes in sensor networks are small, normally 64 bits, so that the encryption or integrity tests do not consume too much energy. Small key sizes lead to the need to update keys on a regular basis. Another aspect of a heterogeneous sensor network is that different sensors measure different environmental data. There are also sensors that can measure more than one environmental phenomenon.

It was shown, in the previous chapter, that if two body sensors sharing no previous secret need to create a shared key, then a password key establishment protocol can be utilised, such as the EKE Protocol [22]. Instead of a password, cryptographically weak physiological data was used. The EKE protocol provides both forward and backwards secrecy.

Forward secrecy specifies that if an adversary obtains the physiological data used to generate the shared key cannot derive the shared key. Backwards secrecy specifies that if an adversary obtains the shared key cannot derive the physiological data. However, the proposed protocol assumed a single SEV value was used to generate the low entropy key. When using SEV data points to create the low entropy keys, the SEV data points are concatenated together before been hashed, as shown in Equation (6.3).

\[
\Psi = H(SEV_1, SEV_2, \ldots, SEV_n) \tag{6.3}
\]

Where \(H(x)\) is a one-way hash function applied to \(x\). The EKE protocol modified to use \(\Psi\) is shown in Protocol 6.4.

The EKE protocol contains a total of four messages. In the context of body sensors, the body sensor \(A\) sends the first message to the body sensor \(B\), the message contains the location of the body sensor \(A\) (the location value is in the clear), and the first part of Diffie–Hellman, \(t_A\), is encrypted by the low entropy key \(\Psi\). It is assumed that both parties can
calculate the exact same SEV and hence the same $\Psi$ values. After the first message is sent, the body sensor $B$ will calculate the second part of the Diffie–Hellman scheme and hence be able to calculate the session key $K_{AB}$. The body sensor $B$ then sends the second part of the Diffie–Hellman scheme encrypted by the low entropy key $\Psi$ to the body sensor $A$. The nonce $n_B$ is also sent, encrypted by the session key $K_{AB}$.

The last two messages authenticate both the body sensors $A$ and $B$, as well as confirming that they have the session key $K_{AB}$. The encryption of the values $t_A$, $t_B$, $n_A$, and $n_B$ can be implemented with an exclusive-or function, as originally described by [22]. The size of $p$ only needs to be 160 bits, which makes it suitable in a resource limited body sensor networks where there is a requirement for the RSA exponent to be small.

The security analysis of this protocol is described in the previous, however, there are differences in the way the SEV is used. The SEV used in this protocol can be over a number of measurements. In the previous chapter an analysis of using IPI for the SEV was not completed. The next section will supply an analysis of IPIs for this protocol.

6.7.2 Analysis of Entropy of Inter-Pulse Intervals

The next set of graphs supplies detailed information and distributions of the IPI values of 54 people where each person was measured over 24 hours, against the proposed SEV-based protocol. A variation of distributions of IPIs over a day will be described. The theoretical amount of entropy for IPIs for a person will also be calculated. A number of worst case scenarios are also described, where it is assumed that the distribution of IPIs will remain the same for the entire population from one day to the next. Also, another worst case scenario is that the distribution of IPIs is known by an adversary and the distribution does not change from one day to the next.
6.7.2.1 Analysis of Different Inter-Pulse Interval Distributions

ECG signals have been examined as a mechanism to create cryptographically strong 128 bit keys [186]. However, the amount of time it takes to collect enough ECG signals to generate a cryptographically strong key may be a hindrance to the adoption of such a mechanism. Also, as more data is used to generate a key, the higher the chance that errors may accumulate from each reading. An examination of the IPI intervals of individuals will be performed to determine their suitability for low entropy keys.

To examine how many beats are required to produce a low entropy key, beat annotation files (about 24 hours each) are used from 54 subjects in normal sinus rhythm from the PhysioNet database [67]. The IPIs were found by finding the time between each R value, sometimes called the RR value. Individuals had varying ranges and distributions of IPI values. Figure 6.7 is some examples of the different distributions found in four individuals. Person A has a lower peak density to the other individuals where the wide range of values has a similar probability of occurring. Whereas, Person B distribution is shifted to the right, and the probability of a smaller range of values of occurring is higher. Person C has more of binomial distribution, and the two peaks have similar probability of occurring. Person D distribution is shifted to the right, and the probability over a smaller range of values of occurring is higher.

The distributions of each individual are unique, where no two distributions are the same. The average mean value of each individual ranged between 0.79 seconds to 1.33 seconds. The standard deviation of each individual has a range between the value of 11 and 33. There was no evidence of any correlation between the mean and the standard deviation. For instance, if someone had a low mean average that does not necessarily result in that person having a low or high standard deviation. The next step is to discover the amount of entropy found in each individual's IPIs.

A summary, where a description is supplied of the smallest observation (Min.), lower quartile (1st Qu.), median, upper quartile (3rd Qu.), and largest observation (Max.), of the four people is supplied in Table 6.2.

<table>
<thead>
<tr>
<th>Person</th>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>53</td>
<td>91</td>
<td>108</td>
<td>108.9</td>
<td>127</td>
<td>181</td>
</tr>
<tr>
<td>B</td>
<td>38</td>
<td>87</td>
<td>108</td>
<td>103.4</td>
<td>120</td>
<td>183</td>
</tr>
<tr>
<td>C</td>
<td>37</td>
<td>84</td>
<td>96</td>
<td>108.2</td>
<td>126</td>
<td>199</td>
</tr>
<tr>
<td>D</td>
<td>52</td>
<td>77</td>
<td>93</td>
<td>94.8</td>
<td>114</td>
<td>141</td>
</tr>
</tbody>
</table>
Figure 6.7: The Different Distributions for Different People
6.7.2.2 Amount of Entropy in a Single Day

Information $I(S_i)$ conveyed by a symbol $S_i$ is inversely proportional to its probability of occurrence $p(S_i)$ [26]. To calculate the binary bits of information contained by a symbol following formula can be used $I(S_i) = \log_2[1/p(S_i)]$ bits (binary unit). The average amount of information per symbol is called the entropy of the source, $H(S)$. For a discrete memoryless source the following formula can be used $H(S) = \sum p(S_i)I(S_i)$ bits/symbol.

On average there is 107,231 IPIs per person every 24 hours. If every IPI had a unique value, then the entropy on average will be 16.7 bits/symbol, which is significantly more than a four digit PIN. However, this maximum value can only be obtained if each IPI value is unique, and depending on the accuracy of the instrument measuring the IPI, there may be some values that may reoccur many times. A mechanism that is used to circumvent this problem is to concatenate IPI values, or to find the entropy of a sequence of values. For instance, for a series of IPI values $\{x_1, x_2, \ldots, x_n\}$, a concatenation with length two values can be created, such as $\{x_1x_2, x_2x_3, \ldots, x_{n-1}x_n\}$. Figure 6.8 shows the distribution of entropy values as values are concatenated from one up to eight concatenated IPI values. When there is a concatenation of four IPI values, the entropy is around 14 bits, which is similar to a four digit PIN. Moving above the four concatenations, the distributions of entropy values move closer to the maximum possible entropy. When examining the entropy, every possible value that can occur is not examined, only the values that actually occurred during the 24 hours for that individual will be examined. The distribution of the entropy is displayed in a boxplot, where the top of the box is the 75 percentile, the bottom of the box is the 25 percentile, and the line in the middle is the mean. Attached to the box are whiskers showing the distribution of other values, with outliers shown as circles. A summary of the eight concatenations is supplied in Table 6.3. At eight concatenations the mean is close to the theoretical limit for the mean. Once there are five concatenations the increase in entropy is not significant. Concatenating a small number of IPI values can lead to a dramatic increase in the entropy, which will then level out after five concatenations.

<table>
<thead>
<tr>
<th>Concat.</th>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.54</td>
<td>5.90</td>
<td>6.08</td>
<td>6.07</td>
<td>6.21</td>
<td>6.72</td>
</tr>
<tr>
<td>2</td>
<td>8.55</td>
<td>9.10</td>
<td>9.45</td>
<td>9.55</td>
<td>10.02</td>
<td>11.65</td>
</tr>
<tr>
<td>3</td>
<td>11.17</td>
<td>12.04</td>
<td>12.51</td>
<td>12.60</td>
<td>13.21</td>
<td>14.51</td>
</tr>
<tr>
<td>4</td>
<td>13.49</td>
<td>14.35</td>
<td>14.78</td>
<td>14.82</td>
<td>15.29</td>
<td>16.07</td>
</tr>
<tr>
<td>5</td>
<td>15.23</td>
<td>15.75</td>
<td>16.01</td>
<td>16.01</td>
<td>16.28</td>
<td>16.58</td>
</tr>
<tr>
<td>7</td>
<td>16.22</td>
<td>16.56</td>
<td>16.65</td>
<td>16.64</td>
<td>16.71</td>
<td>16.98</td>
</tr>
</tbody>
</table>
Figure 6.8: The Different Distributions of the Entropy with IPI Values Concatenated
**CHAPTER 6. VALIDATION**

### 6.7.2.3 Worst Case Scenario When Using Entire Population

Finding the entropy of the data is a good indication of the complexity found in the data, but it should not be solely used to find how cryptographically secure the data is. If an assumption is made that the 24 hours worth of IPI values obtained from the 54 individuals will be the same day after day, and another assumption is that an adversary has obtained all the IPI values the probability can be obtained for an adversary able to guess the correct IPI value. It should be noted that the assumption that the IPI values are the same day after day is done only to simplify the problem and to do a worst case analysis.

Figure 6.9 gives the distribution of the probability of success if an adversary chooses the most likely IPI value for the entire population and attacks each of the 54 individuals. As before, distributions for one concatenated IPI value up to eight concatenated IPI values is given. The distributions are shown as a boxplot, but a map of the probability points in each of the distributions is also supplied.

The probabilities decrease exponentially, and once four IPI values are concatenated two of the individuals will never have the sequence. When eight IPI values are concatenated there will be 74% of the individuals who will never have that sequence. When only one IPI value is used, there are probabilities as high as 4e-02, however, there is one as low as 2e-04. To correctly guess a four digit PIN, has a probability of 1e-04.

A summary, where a description is supplied of the smallest observation (Min.), lower quartile (1st Qu.), median, upper quartile (3rd Qu.), largest observation (Max.), and the people who have zero probability are marked as NA, of the eight concatenations is supplied in Table 6.4.

**Table 6.4: Summary Information using the Most Likely Result for the Low Entropy Key**

<table>
<thead>
<tr>
<th>Concat.</th>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
<th>NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.2e-04</td>
<td>1.4e-02</td>
<td>1.9e-02</td>
<td>2.0e-02</td>
<td>2.4e-02</td>
<td>4.7e-02</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>9.8e-05</td>
<td>2.8e-03</td>
<td>4.0e-03</td>
<td>4.4e-03</td>
<td>5.8e-03</td>
<td>1.2e-02</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1.2e-05</td>
<td>5.4e-04</td>
<td>1.0e-03</td>
<td>1.3e-03</td>
<td>1.9e-03</td>
<td>5.4e-03</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4.0e-05</td>
<td>1.6e-04</td>
<td>3.3e-04</td>
<td>5.0e-04</td>
<td>8.0e-04</td>
<td>2.4e-03</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1.0e-05</td>
<td>5.4e-05</td>
<td>1.2e-04</td>
<td>2.2e-04</td>
<td>3.5e-04</td>
<td>1.1e-03</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>7.0e-06</td>
<td>2.1e-05</td>
<td>5.2e-05</td>
<td>1.0e-04</td>
<td>1.5e-04</td>
<td>5.2e-04</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>8.0e-06</td>
<td>1.8e-05</td>
<td>4.1e-05</td>
<td>5.4e-05</td>
<td>5.9e-05</td>
<td>2.1e-04</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>1.0e-05</td>
<td>1.0e-05</td>
<td>2.0e-05</td>
<td>5.0e-05</td>
<td>5.0e-05</td>
<td>3.0e-04</td>
<td>40</td>
</tr>
</tbody>
</table>

At four concatenations there are two people who never get the most common IPI sequence of values. The probability of a guess after five concatenations does not increase dramatically; however, there is a dramatic increase in the number of people who never get the most common IPI sequence during a day.
Figure 6.9: Using the Most Likely Result for the Low Entropy Key
6.7.2.4 Worst Case Scenario using Individual Distributions

If another assumption is made that the adversary knows each individual’s IPI distribution, and their distribution will stay the same day after day, to give an adversary more information. This assumption is highly unlikely but is done for simplification as a worst case scenario.

Figure 6.10 gives the distribution of the probability of success if an adversary chooses the most likely IPI value for that individual. As before, distributions were taken for one concatenated IPI value up to eight concatenated IPI values. The distributions are shown as a boxplot, but the map of probability points in each of the distributions is also displayed.

The probabilities decrease exponentially, and once the protocols use four IPIs concatenated together, the protocols start to get the majority of the individuals having probabilities similar to a PIN. If the individual’s distribution changes over the days, then the probabilities may be different.

A summary, where a description is supplied of the smallest observation (Min.), lower quartile (1st Qu.), median, upper quartile (3rd Qu.), and largest observation (Max.), of the eight concatenations is supplied in Table 6.5.

<table>
<thead>
<tr>
<th>Concat.</th>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.6e-02</td>
<td>2.5e-02</td>
<td>2.8e-02</td>
<td>3.0e-02</td>
<td>3.3e-02</td>
<td>4.9e-02</td>
</tr>
<tr>
<td>2</td>
<td>2.2e-03</td>
<td>4.7e-03</td>
<td>6.3e-03</td>
<td>7.0e-03</td>
<td>8.3e-03</td>
<td>1.6e-02</td>
</tr>
<tr>
<td>3</td>
<td>4.5e-04</td>
<td>1.3e-03</td>
<td>1.9e-03</td>
<td>2.2e-03</td>
<td>2.8e-03</td>
<td>6.8e-03</td>
</tr>
<tr>
<td>4</td>
<td>1.2e-04</td>
<td>4.0e-04</td>
<td>5.5e-04</td>
<td>8.3e-04</td>
<td>1.1e-03</td>
<td>3.3e-03</td>
</tr>
<tr>
<td>5</td>
<td>4.9e-05</td>
<td>1.5e-04</td>
<td>2.2e-04</td>
<td>3.8e-04</td>
<td>5.2e-04</td>
<td>1.7e-03</td>
</tr>
<tr>
<td>6</td>
<td>2.9e-05</td>
<td>7.4e-05</td>
<td>1.1e-04</td>
<td>1.9e-04</td>
<td>2.4e-04</td>
<td>9.4e-04</td>
</tr>
<tr>
<td>7</td>
<td>2.4e-05</td>
<td>4.2e-05</td>
<td>6.4e-05</td>
<td>1.1e-04</td>
<td>1.2e-04</td>
<td>5.1e-04</td>
</tr>
<tr>
<td>8</td>
<td>1.8e-05</td>
<td>3.0e-05</td>
<td>3.7e-05</td>
<td>6.5e-05</td>
<td>7.1e-05</td>
<td>3.0e-04</td>
</tr>
</tbody>
</table>

At eight concatenations the mean is close to the theoretical limit for the mean. Once there are five concatenations the increase in entropy is not significant. Concatenating a small number of IPI values can lead to a dramatic increase in the entropy, which will then level out after five concatenations. If the assumption is made that every IPI value is unique during a day, then the lowest value will be 9.3e-06. The values in the table are converging towards the lowest theoretical value, and once there are eight concatenations the values are less than an order of magnitude away.
Figure 6.10: The Different Distributions after Concatenating IPI Values
6.7.3 Problems with using SEVs with Error Correction

The limitation of the above protocol is that there is no margin for error when specifying the value for $\Psi$. The physiological data measured by its very nature has a margin of error dependent on the instrument used to measure the data. Values from two instruments may be out by that margin of error. If the value is out by even one bit, the above protocol will fail. An extensive search of the literature found that secure key establishment protocols do not incorporate any error margin in the shared secret between two parties. For instance, if a certificate, symmetric key, or password is out by one bit, then the key establishment protocol will fail.

A new type of protocol is proposed, called Error Correcting Key Establishment (ECKE) protocol. The ECKE protocol establishes a key by incorporating the error-margin when sensing the $\Psi$ from the sensor. An example of an ECKE protocol is described in Protocol 6.5.

**Protocol 6.5** ECKE — EKE based protocol with error correction

<table>
<thead>
<tr>
<th>Shared Information:</th>
<th>Generator $g$ of $G$ where $p - 1 = qr$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$B$</td>
</tr>
<tr>
<td>Read $\Psi_i$</td>
<td>Read $\Psi_j$</td>
</tr>
<tr>
<td>$r_A \in R \mathbb{Z}_p$</td>
<td>$r_B \in R \mathbb{Z}_p$</td>
</tr>
<tr>
<td>$t_A = g^{r_A}$</td>
<td>$K_j = GenK(\Psi_j, r_B, t_A \oplus \Psi_i)$</td>
</tr>
<tr>
<td>$t_A \oplus \Psi_i$</td>
<td>$t_B = g^{r_B}$</td>
</tr>
<tr>
<td>$x = GenX(n_B, K_j)$</td>
<td></td>
</tr>
<tr>
<td>$K_i = GenK(\Psi_i, r_A, t_B \oplus \Psi_j)$</td>
<td></td>
</tr>
<tr>
<td>$y = GenY(n_A, x, K_i)$</td>
<td>${K_{AB}, n_A} = V_B(y, K_j)$</td>
</tr>
<tr>
<td>$K_{AB} = V_A(n_A \oplus K_{AB}, K_j)$</td>
<td>$n_A \oplus K_{AB}$</td>
</tr>
</tbody>
</table>

The sensor node $A$ at the start of the protocol reads the physiological data, reads $n$ values and produces a one-way hash of the concatenation to produce $\Psi_i$, as shown in Equation (6.3). At the same time the sensor node $B$ will produce $\Psi_j$, there will be the same number of concatenations, and the same one-way hash function will be used. The sensor node $A$ will create a random number $r_A$ and then produce the Diffie–Hellman component $t_A = g^{r_A}$. The sensor node $A$ will then send the Diffie–Hellman component $t_A \oplus \Psi_i$ to the sensor node $B$. The symbol $\oplus$ denotes encryption, and may be as simple as an exclusive-or function. Encryption of data using a weak key introduces a brute force style attack in this protocol. An adversary guessing the low entropy key $\Psi_i$ can attempt to decrypt $t_A \oplus \Psi_i$ and examine whether the resulting plaintext is a valid Diffie–Hellman ephemeral value. For instance, if the decryption with a candidate $\Psi$ results in a string whose value is greater...
than $p$, then that candidate is invalid. By choosing $p$ wisely, such that the value of $p$ is moved further away from $2^{N-1} + 1$, the security can be strengthened significantly, and the time-frame to remove a significant number of low entropy keys is large enough so that the $\Psi$ should have changed in that period. The technique is used in password protocols where RSA is used [22].

When the sensor node $B$ receives the message it will create a random number $r_B$. After $r_B$ is created, the sensor node $B$ will generate a set of possible keys $K_j$ as defined in Equation (6.4).

$$K_j = \text{GenK}(\Psi_j, r_B, t_A \oplus \Psi_i) = K_j = \{K_{1AB}, K_{2AB}, \ldots, K_{mAB}\} \quad (6.4)$$

The cardinality of the set $K_j$ is defined as $|K_j| = 3^n$.

In the case where $n = 1$, the sensor node $B$ will then generate three possible keys, $K_{1AB}, K_{2AB}, K_{3AB}$, as shown in Equation (6.5).

$$K_{1AB} = (t_A \oplus \Psi_i \oplus (\Psi_j - E))^r_B$$
$$K_{2AB} = (t_A \oplus \Psi_i \oplus (\Psi_j))^r_B$$
$$K_{3AB} = (t_A \oplus \Psi_i \oplus (\Psi_j + E))^r_B \quad (6.5)$$

The value of $E$ is the increment (step size) to the next valid value. For instance, if the SEV values were given in increments of 0.1 seconds, the value of $E$ would be 0.1. It is assumed that for the security protocol, the possible values of for the SEV are rounded to a discrete level where the margin of error is taken into consideration. So if two nodes where reading the same physiological data, then the three possible values can be obtained from $S \pm E$.

The sensor node $B$ calculates three possible values for the shared symmetric key between the two sensors. If $\Psi_i = \Psi_j - E$, then the calculation for $K_{1AB}$ can be simplified to $K_{1AB} = t_A^r_B$. If $\Psi_i = \Psi_j$, then the calculation for $K_{2AB}$ can be simplified to $K_{2AB} = t_A^r_B$. If $\Psi_i = \Psi_j + E$, then the calculation for $K_{3AB}$ can be simplified to $K_{3AB} = t_A^r_B$.

The sensor node $B$ also creates a nonce $n_B$, the nonce is encrypted with the possible key values. The same nonce can be encrypted by each key safely, since a property of the security keys is that they are independent of each other. The value of $x$ is defined in Equation (6.6).

$$x = n_B \oplus K_{1AB}, n_B \oplus K_{2AB}, \ldots, n_B \oplus K_{mAB} \quad (6.6)$$

The Diffie–Hellman component $t_B = g^x$ is encrypted and sent by the sensor node $B$ to the sensor node $A$. 

---

183
In the case where \( n = 1 \) the message that will be created is described in Equation (6.7).

\[
t_B \oplus \Psi_j, n_B \oplus K_{1AB}, n_B \oplus K_{2AB}, n_B \oplus K_{3AB} \quad (6.7)
\]

When the sensor node \( A \) receives the message it will also generate the possible keys, as described in Equation (6.8).

\[
K_i = GenK(\Psi_i, r_A, t_B \oplus \Psi_j) = \{K_{1BA}, K_{2BA}, \ldots, K_{mBA}\} \quad (6.8)
\]

When \( n = 1 \) sensor node \( A \) will generate three possible keys, \( K_{1BA}, K_{2BA}, K_{3BA} \), as shown in Equation (6.9).

\[
K_{1BA} = (t_B \oplus \Psi_j \oplus (\Psi_i - E))^r_A
K_{2BA} = (t_B \oplus \Psi_j \oplus (\Psi_i))^r_A
K_{3BA} = (t_B \oplus \Psi_j \oplus (\Psi_j + E))^r_A \quad (6.9)
\]

The sensor node \( A \) calculates three possible values for the shared symmetric key between the two sensors. In the Diffie–Hellman protocol the shared key is normally calculated as

\[
K_{AB} = t_A^r_B = t_B^r_A \quad (6.10)
\]

Once the possible keys are created an attempt to decrypt the nonce \( n_B \) is performed. There are \( m \) possible nonces, and \( m \) possible keys. There are \( m^2 \) possible values. The value of \( y \) is calculated.

\[
y = (n_A, n_{1B}, \ldots, n_{m^2B}) \oplus K_{1BA},
(n_A, n_{1B}, \ldots, n_{m^2B}) \oplus K_{2BA}, \ldots,
(n_A, n_{1B}, \ldots, n_{m^2B}) \oplus K_{mBA} \quad (6.11)
\]

For the case where \( n = 1 \) the sensor node \( A \) will derive nine possible values for the nonce \( n_B \). The nine possible nonce values are due to the nine different decryption executions to obtain the nonce \( n_B \). The value \( n_B \oplus K_{1AB} \) is decrypted using the three keys \( K_{1BA}, K_{2BA} \) and \( K_{3BA} \), to get three possible nonces \( n_{1B}, n_{2B} \) and \( n_{3B} \). Six more possible nonces are decrypted from \( n_B \oplus K_{2AB} \) and \( n_B \oplus K_{3AB} \). At this stage the sensor node \( A \) does not know the correct key to use for communication. The sensor node \( A \) will create a nonce \( n_A \), the nonce and the nine possible \( n_B \) values are encrypted with the three possible key values \( K_{1BA}, K_{2BA} \) and \( K_{3BA} \).
6.7. VALIDATING INTER-PULSE INTERVALS

\[ y = (n_A, n_1B \ldots n_9B) \oplus K_{B1A}, (n_A, n_1B \ldots n_9B) \oplus K_{2BA}, (n_A, n_1B \ldots n_9B) \oplus K_{3BA} \] (6.12)

The sensor node B receives the encrypted nonces and uses all of the keys \( K_j \), to decrypt each component of the message. There are \( |K_j| \) decryption calculations performed. Only one of them will contain a valid value for \( n_B \), when the valid value for \( n_B \) is found, the key that is decrypted is the shared secret between both the sensors. When the valid value of \( n_B \) is found, the value of \( n_A \) is also discovered by the sensor node B, as shown in Equation (6.13). The sensor node B will encrypt the nonce \( n_A \) with the discovered shared secret.

\[ \{K_{AB}, n_A\} = V_B(y, K_j) \] (6.13)

When the sensor node A receives the last message, it will decrypt the message using all the possible keys \( K_i \). Whichever key was able to successfully decrypt the message is considered to be the correct key. If \( \Psi_i = \Psi_j + E \), then \( K_{AB} = K_{3AB} = K_{1BA} \). If \( \Psi_i \neq \Psi_j \) and \( \Psi_i \neq \Psi_j \pm E \), for instance, from environmental noise or from some other form of interference causing a misreading, then the protocol will fail to establish a key, and the protocol will need to be rerun.

\[ K_{AB} = V_A(n_A \oplus K_{AB}, K_i) \] (6.14)

In this section, a protocol has been defined that handles two sensors reading the same physiological data with a level of uncertainty in the data. The protocol does increase the communication, when more physiological data is added to increase the entropy.

6.7.4 Analysis of Inter-Pulse Intervals against Error Correcting Protocols

The next set of graphs supplies detailed information and distributions of the IPI values of 54 people where each person was measured over 24 hours, against the proposed error correcting SEV-based protocol. The low entropy key is generated using several different methods, to attempt to get more randomness into the key. The methods are based on the number of the same IPIs found in the concatenation. The three different methods are:

- Every sequence of IPIs is allowed to be the low entropy key.
- Every sequence of IPIs is allowed except where every IPI value is the same.
- Every sequence of IPIs is allowed except where more than half IPI values are the same.
6.7.4.1 Worst Case Scenario using Individual Distributions

To analyse if the error-correcting protocol is suitable for an individual, the probability that an adversary can successfully guess the low entropy key will be found. If an assumption is made that the adversary knows each individual’s IPI distribution and their distribution will stay the same day after day. As described before, this assumption may not be correct but is done for simplification.

Figure 6.11 gives the distribution of the probability of success if an adversary chooses the most likely IPI value for that individual, and the devices take the error margin into account. As before, distributions for one concatenated value up to eight concatenated IPI values is supplied. The distributions are shown as a boxplot, but a map of the probability points in each of the distributions is also displayed.

Some of the probabilities decrease exponentially, however, the majority of the probabilities decrease at a rate where even after eight concatenations, and the majority of the probabilities are significantly more than the probabilities found in guessing a four digit PIN.

A summary, where a description is supplied of the smallest observation (Min.), lower quartile (1st Qu.), median, upper quartile (3rd Qu.), and largest observation (Max.), of the eight concatenations is supplied in Table 6.6.

<table>
<thead>
<tr>
<th>Concat.</th>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.8e-02</td>
<td>7.2e-02</td>
<td>8.0e-02</td>
<td>8.5e-02</td>
<td>9.5e-02</td>
<td>1.4e-01</td>
</tr>
<tr>
<td>2</td>
<td>1.7e-02</td>
<td>3.3e-02</td>
<td>4.2e-02</td>
<td>4.2e-02</td>
<td>4.9e-02</td>
<td>9.0e-02</td>
</tr>
<tr>
<td>3</td>
<td>6.2e-03</td>
<td>1.6e-02</td>
<td>2.4e-02</td>
<td>2.5e-02</td>
<td>3.2e-02</td>
<td>6.2e-02</td>
</tr>
<tr>
<td>4</td>
<td>2.5e-03</td>
<td>8.8e-03</td>
<td>1.5e-02</td>
<td>1.6e-02</td>
<td>2.1e-02</td>
<td>4.3e-02</td>
</tr>
<tr>
<td>5</td>
<td>8.1e-04</td>
<td>4.6e-03</td>
<td>8.1e-03</td>
<td>1.0e-02</td>
<td>1.4e-02</td>
<td>3.1e-02</td>
</tr>
<tr>
<td>6</td>
<td>3.3e-04</td>
<td>2.9e-03</td>
<td>4.9e-03</td>
<td>6.9e-03</td>
<td>9.5e-03</td>
<td>2.5e-02</td>
</tr>
<tr>
<td>7</td>
<td>4.9e-05</td>
<td>1.8e-03</td>
<td>2.8e-03</td>
<td>4.8e-03</td>
<td>6.8e-03</td>
<td>1.9e-02</td>
</tr>
<tr>
<td>8</td>
<td>3.7e-05</td>
<td>8.5e-04</td>
<td>2.0e-03</td>
<td>3.2e-03</td>
<td>4.1e-03</td>
<td>1.6e-02</td>
</tr>
</tbody>
</table>

If an assumption is made that every IPI value is unique during a day, and the lowest probabilistic value possible is 9.3e-06. Even after eight concatenations, there are still orders of magnitude away from the lowest theoretical value. The maximum value has not significantly decreased between one to eight concatenations. With further investigation, it was found that the most frequent IPI sequences are when all the IPI values are the same in the sequence. In the next section any sequence where the IPI values are the same will be removed.
6.7. VALIDATING INTER-PULSE INTERVALS

Figure 6.11: The Different Distributions after Concatenating IPI Values with the Error Margin
6.7.4.2 Worst Case Scenario Excluding Same Inter-Pulse Intervals

Some of the more common most likely concatenated IPI sequence is when all the IPI values within that sequence are the same. If the requirement is added that there cannot be a concatenated IPI sequence where all the IPI values are the same. Figure 6.12 gives the distribution of the probability of success if an adversary chooses the most likely IPI value for that individual, and the devices take the error margin into account. As before, distributions are taken for one concatenated IPI value up to eight concatenated IPI values. The distributions are shown as a boxplot, but also a map of the probability points in each of the distributions is displayed.

Some of the probabilities decrease exponentially; however, there is no major increase in the security of the system. The majority of the probabilities decrease at a rate where even after eight concatenations; the majority of the probabilities are significantly more than the probabilities found in guessing a four digit PIN.

A summary, where a description is supplied of the smallest observation (Min.), lower quartile (1st Qu.), median, upper quartile (3rd Qu.), and largest observation (Max.), of the eight concatenations is supplied in Table 6.7.

Table 6.7: Summary Information Concatenating IPI Values that are Different with the Error Margin

<table>
<thead>
<tr>
<th>Concat.</th>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.8e-02</td>
<td>7.2e-02</td>
<td>8.0e-02</td>
<td>8.5e-02</td>
<td>9.5e-02</td>
<td>1.4e-01</td>
</tr>
<tr>
<td>2</td>
<td>1.5e-02</td>
<td>2.6e-02</td>
<td>3.3e-02</td>
<td>3.3e-02</td>
<td>3.8e-02</td>
<td>6.8e-03</td>
</tr>
<tr>
<td>3</td>
<td>6.1e-03</td>
<td>1.5e-02</td>
<td>2.1e-02</td>
<td>2.1e-02</td>
<td>2.5e-02</td>
<td>4.9e-02</td>
</tr>
<tr>
<td>4</td>
<td>2.4e-02</td>
<td>9.0e-03</td>
<td>1.3e-02</td>
<td>1.4e-02</td>
<td>1.7e-02</td>
<td>3.1e-02</td>
</tr>
<tr>
<td>5</td>
<td>8.1e-04</td>
<td>4.7e-03</td>
<td>8.3e-03</td>
<td>9.3e-03</td>
<td>1.3e-02</td>
<td>2.4e-02</td>
</tr>
<tr>
<td>6</td>
<td>3.0e-04</td>
<td>2.8e-03</td>
<td>5.4e-03</td>
<td>6.2e-03</td>
<td>9.2e-03</td>
<td>1.8e-02</td>
</tr>
<tr>
<td>7</td>
<td>4.9e-05</td>
<td>9.8e-04</td>
<td>3.2e-03</td>
<td>4.3e-03</td>
<td>5.9e-03</td>
<td>1.5e-02</td>
</tr>
<tr>
<td>8</td>
<td>2.1e-05</td>
<td>4.7e-04</td>
<td>1.7e-03</td>
<td>2.5e-03</td>
<td>3.7e-03</td>
<td>1.2e-02</td>
</tr>
</tbody>
</table>

If an assumption is made that every IPI value is unique during a day, then the lowest probabilistic value is 9.3e-06. The values a slightly better than when the sequence of the same IPI values were kept. However, there was not a significant (orders of magnitude) improvement in the values. Even after eight concatenations, there are still orders of magnitude away from the lowest theoretical value. The maximum value has not significantly decreased between one to eight concatenations. The most frequent sequence of values now seems to contain values where the majority of the IPI values are the same. In the next section any sequence where more than half the values are the same will be removed.
Figure 6.12: The Distributions after Concatenating IPI Values that are Different with the Error Margin
6.7.4.3 Worst Case Scenario Excluding Half of the Same Inter-Pulse Intervals

The next set of some of the more common most likely concatenated IPI sequence is when half or more of the IPI values within that sequence are the same. If the requirement is added that there cannot be a concatenated IPI sequence where half or more of the IPI values are the same. Figure 6.13 gives the distribution of the probability of success if an adversary chooses the most likely IPI value for that individual, and the devices take the error margin into account. As before, distributions are taken for one concatenated IPI value up to eight concatenated IPI values. The distributions are shown as a boxplot, but also a map of the probability points in each of the distributions is displayed.

Some of the probabilities decrease exponentially; however, there is no major increase in the security of the system. The majority of the probabilities decrease at a rate where even after eight concatenations; the majority of the probabilities are significantly more than the probabilities found in guessing a four digit PIN.

A summary, where a description is supplied of the smallest observation (Min.), lower quartile (1st Qu.), median, upper quartile (3rd Qu.), and largest observation (Max.), of the eight concatenations is supplied in Table 6.8.

<table>
<thead>
<tr>
<th>Concat.</th>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.8e-02</td>
<td>7.2e-02</td>
<td>8.0e-02</td>
<td>8.4e-02</td>
<td>9.4e-02</td>
<td>1.4e-01</td>
</tr>
<tr>
<td>2</td>
<td>1.5e-02</td>
<td>2.6e-02</td>
<td>3.3e-02</td>
<td>3.3e-02</td>
<td>3.8e-02</td>
<td>6.8e-02</td>
</tr>
<tr>
<td>3</td>
<td>4.2e-03</td>
<td>9.1e-03</td>
<td>1.3e-02</td>
<td>1.2e-02</td>
<td>1.5e-02</td>
<td>2.4e-02</td>
</tr>
<tr>
<td>4</td>
<td>1.2e-03</td>
<td>6.4e-03</td>
<td>9.0e-03</td>
<td>9.5e-03</td>
<td>1.1e-02</td>
<td>2.4e-02</td>
</tr>
<tr>
<td>5</td>
<td>7.2e-04</td>
<td>2.5e-03</td>
<td>3.6e-03</td>
<td>4.2e-03</td>
<td>5.7e-03</td>
<td>1.3e-02</td>
</tr>
<tr>
<td>6</td>
<td>3.1e-04</td>
<td>1.6e-03</td>
<td>3.2e-03</td>
<td>3.7e-03</td>
<td>4.7e-03</td>
<td>1.3e-02</td>
</tr>
<tr>
<td>7</td>
<td>3.2e-05</td>
<td>4.5e-04</td>
<td>1.3e-03</td>
<td>1.8e-03</td>
<td>2.4e-03</td>
<td>7.5e-03</td>
</tr>
<tr>
<td>8</td>
<td>2.1e-05</td>
<td>3.8e-04</td>
<td>1.0e-03</td>
<td>1.4e-03</td>
<td>1.8e-03</td>
<td>6.3e-03</td>
</tr>
</tbody>
</table>

If assumption is made that every IPI value is unique during a day, then the lowest probabilistic value is 9.3e-06. There was some improvement, especially around the maximum values. However, there was no significant (orders of magnitude) change in the values. Even after eight concatenations, there are still orders of magnitude away from the lowest theoretical value. By including the uncertainty of the readings into the proposed protocols, any benefits of excluding frequent sequences has been removed. The dramatic increase in communication costs and the large decrease in security have limited the use of these types of protocols. A new method is required where the security does not decrease, and there is no dramatic increase in communication is required.
Figure 6.13: The Distributions after Concatenating Less Than Half the Same IPI Values with the Error Margin
6.7.5 Protocol with Communication Overhead Removed

The concatenation of the IPI values dramatically increases the message sizes in the protocol, and once the error margin is taken into consideration, the security of the protocol does not increase as dramatically as before. Another type of protocol may be more useful for concatenated IPI values. The Anderson–Lomas protocol [7] is a unique password protocol that uses a collisionful hash function as shown in Protocol 6.6.

**Protocol 6.6 Anderson–Lomas protocol**

$A$ and $B$ both know $\Psi$.

$A$ creates $t_A = g^{r_A}$, where $r_A$ is a random value.

$B$ creates $t_B = g^{r_B}$, where $r_B$ is a random value.

$K_{AB} = g^{r_A r_B}$

$M_1 \ A \rightarrow B : \ t_A$

$M_2 \ B \rightarrow A : \ t_B$

$M_3 \ A \rightarrow B : \ h_{\Psi}(K_{AB})$

$M_4 \ B \rightarrow A : \ h_{H(\Psi)}(K_{AB})$

As shown in Protocol 6.6, the first two messages are the Diffie–Hellman part of the protocol, whereas the second two messages authenticate that $A$ sent $t_A$ and that $B$ sent $t_B$.

The function $h$ is a collisionful hash function. The function is supplied with two parameters, a key $k$ and a bit-string $x$. When function $h$ is supplied with $k$ and $x$ it is easy to compute $h_k(x)$. If function $h$ is supplied with $k$ and $h_k(x)$ it is hard to find a value $y$ such that $h_k(x) = h_k(y)$ but $x$ does not equal $y$. Given $x$ and $h_k(x)$ it is hard to compute $k$, though it is less hard to find $k'$ such that $h_k(x) = h_{k'}(x)$, where $k$ does not equal $k'$. The specific collisionful hash function used in the Anderson–Lomas protocol is:

$$h_k(x) = H(MAC_k(x) \mod 2^m, x) \quad (6.15)$$

The value $m$ is a pre-defined value known by all parties and $H$ is a one-way hash function. It is easy to find many $k$ values that give the same output when $m$ is small. Anderson and Lomas suggest to take $m = n/2$ where $2^n$ is the size of the password space, allowing the protocol to use $\Psi$ as the key $h$ and so guess for $\Psi$ can only be verified with probability $2^{-n/2}$.

An adversary can try to mount an active attack by posing as $B$ and attempting to guess the message $M_3$; this will succeed with probability $2^{-n/2}$. Therefore, an active adversary can reduce the number of possible passwords to the square root of the initial number of passwords. Thus in traditional password systems the protocol leaks a larger amount of information compared to other password protocols. A defence against this type of attack is that if an invalid password check has been detected, the password needs to be changed. This is difficult in traditional password systems and can produce an effective denial-of-service attack. However, in an environment where the SEVs are continually changing, this
6.7. VALIDATING INTER-PULSE INTERVALS

A problem occurs when the error margin is taken into consideration. Sensor node B may find multiple collisions in its error margin.

The protocol has been modified slightly to remove one of the messages, and to utilise elliptic curves, as described by the Error Correcting Collisionful Hash (ECCH) protocol, defined in Protocol 6.7. Sensor node A will generate the low entropy key $\Psi$, and sensor node B will generate a set of low entropy keys $\Psi_B$, based upon the SEVs that it sensed, and the error margin in the SEV.

**Protocol 6.7 ECCH — Error Correcting Collisionful Hash Protocol**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>$A \rightarrow B$: $t_A$</td>
</tr>
<tr>
<td>M2</td>
<td>$B \rightarrow A$: $t_B$</td>
</tr>
<tr>
<td>M3</td>
<td>$A \rightarrow B$: $h_\Psi(K_{AB})$</td>
</tr>
<tr>
<td>M4</td>
<td>$B \rightarrow A$: $h_{H(\Psi_1)}(K_{AB}), \ldots, h_{H(\Psi_k)}(K_{AB})$</td>
</tr>
</tbody>
</table>

The first three messages are essentially the same as found in the Anderson–Lomas protocol.

The sensor node A will choose the $\Psi$ and use that to calculate the hash of the key, and sends the value $h_\Psi(K_{AB})$. However, the sensor node B does not know the exact value of $\Psi$. It can be a possible $|\Psi_B|$ values. Sensor node B can determine the value of $\Psi$ by calculating the hash of $K_{AB}$ for each of the possible sensed values. For each of the correct $\Psi$ values that are found, then the sensor node B can successfully send the value $h_\Psi(x_{\Psi}(K_{AB}))$.

Finding valid values to use in this protocol is more difficult, since to calculate the low entropy key space the total number of possible $\Psi$ values is needed for each concatenation. However, if a value, such as four or five, is used for the concatenation, there is not have enough raw data from the 54 data sets to produce a reasonable set of possible values that may occur. The equation to find the total number of possible values is $x(d^{m-1})$, where $x$ is the total number of possible IPIs, $d$ is the distance that the IPI can change from one value to the next, and $m$ is the number of IPIs used to create the low entropy key. To find a suitable number of values that a low entropy key can belong to for healthy people, the amount of IPI change from one reading to the next needs to be found. The **InterQuartile Distance (IQD)** mechanism is used to find the outliers. Using IQD the change that can occur is normally $\pm 0.08$ seconds, with only four percent of changes outside of these values. The IPI for the 54 subjects fell between 30 and 200.

Table 6.9 provides the resultant numbers obtained when concatenating IPIs up to eight. The protocol is tested experimentally over 100000 iterations and obtained similar figures. In the experiments, there was never more than three collisions. Also, it should be noted...
that by using the IQD, a large amount of randomness was removed, and the possible key space could be larger than the experiments had used. If the key space is larger than security of the protocol also improves.

Table 6.9: Concatenated IPI Values from ECG Data for the ECCH Protocol

<table>
<thead>
<tr>
<th>Number of IPIs</th>
<th>Key space size (bits)</th>
<th>modulus value</th>
<th>Probability of guess</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.4</td>
<td>4</td>
<td>6.3e-02</td>
</tr>
<tr>
<td>2</td>
<td>11.4</td>
<td>6</td>
<td>1.6e-02</td>
</tr>
<tr>
<td>3</td>
<td>15.4</td>
<td>8</td>
<td>3.9e-03</td>
</tr>
<tr>
<td>4</td>
<td>19.4</td>
<td>10</td>
<td>9.8e-04</td>
</tr>
<tr>
<td>5</td>
<td>23.4</td>
<td>12</td>
<td>2.4e-04</td>
</tr>
<tr>
<td>6</td>
<td>27.4</td>
<td>14</td>
<td>6.1e-05</td>
</tr>
<tr>
<td>7</td>
<td>31.4</td>
<td>16</td>
<td>1.5e-05</td>
</tr>
<tr>
<td>8</td>
<td>35.4</td>
<td>18</td>
<td>3.8e-06</td>
</tr>
</tbody>
</table>

Table 6.10 provides the resultant numbers obtained when concatenating IPIs up to eight. The protocol is tested experimentally over 100 000 iterations and obtained similar figures. In the experiments, there was never more than three collisions. If the experiments continued to over one million iterations, there should be more collisions, and there would be more likelihood of four collisions. The theoretical probability of having more than four collisions is more than one in a million. Also, it should be noted that by using the IQD, a large amount of randomness was removed, and the possible key space could be larger than the experiments had used. If the key space is larger than security of the protocol also improves.

Table 6.10: Probability of Message Sizes for ECCH Protocol

<table>
<thead>
<tr>
<th>Number of IPIs</th>
<th>Probability of message size greater than one</th>
<th>Probability of message size greater than two</th>
<th>Probability of message size greater than three</th>
<th>Probability of message size greater than four</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.2e-01</td>
<td>4.0e-03</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1.2e-01</td>
<td>6.7e-03</td>
<td>2.2e-04</td>
<td>4.4e-06</td>
</tr>
<tr>
<td>3</td>
<td>9.6e-02</td>
<td>4.6e-03</td>
<td>1.4e-04</td>
<td>3.2e-06</td>
</tr>
<tr>
<td>4</td>
<td>7.5e-02</td>
<td>2.9e-03</td>
<td>7.3e-05</td>
<td>1.4e-06</td>
</tr>
<tr>
<td>5</td>
<td>5.6e-02</td>
<td>1.6e-03</td>
<td>3.1e-05</td>
<td>4.4e-07</td>
</tr>
<tr>
<td>6</td>
<td>4.3e-02</td>
<td>9.6e-04</td>
<td>1.4e-05</td>
<td>1.6e-07</td>
</tr>
<tr>
<td>7</td>
<td>3.2e-02</td>
<td>5.3e-04</td>
<td>5.7e-06</td>
<td>4.7e-08</td>
</tr>
<tr>
<td>8</td>
<td>2.5e-02</td>
<td>3.1e-04</td>
<td>2.5e-06</td>
<td>1.6e-08</td>
</tr>
</tbody>
</table>

Figure 6.14 shows the probability of not getting any extra collisions. Extra hits will cause the communication costs to increase. The experimental results was run over 100 000 iterations. The experimental results and theoretical results are shown to be similar. Also,
as the number of concatenations increase so does the likelihood that there will not be an extra collisions.

![Graph showing the probability of zero collisions against the number of concatenated IPIs.](image)

**Figure 6.14:** The Probability of Not Getting Multiple Collisions using the ECCH

### 6.7.6 Plaintext Attacks

Care must be taken when using data both to establish keys and as plaintext. The SEV is important for the new protocols and also important to diagnose medical problems that may be occurring. A feature of most password protocols is if the password (or low entropy key) is compromised, after the execution of the protocol, then an attacker cannot compromise any session keys created from that protocol run and any previous protocol runs. The SEV can be safely used to diagnose medical conditions that are important to check for the patient.

Another problem is if an attacker obtains the physiological data before the running of the protocol. There has not been any analysis of this type of attack in the literature. To counter this attack the system should be secure against an attacker obtaining any of the SEVs. With this assumption then the adversary does not know the physiological data before the protocol gets run; so they cannot make an educated guess for the SEVs used during a protocol run.

Another important aspects of the protocols is that the protocols do not use the randomness in the body to create the new session keys. The physiological data found on a body is used to authenticate the body. The data is used to ensure that two devices that are
requesting to communicate to each are on the same body. The randomness to generate the Diffie–Hellman components of the protocols is accomplished via other well-known cryptographic mechanisms.

The ECKE protocol was tested using physiological values obtained from an Android smart phone. The implementation of the application is described in the next section.

### 6.7.7 Implementation and Analysis

Video camera technologies in smart phones enable photoplethysmographic acquisition and hence measure inter–pulse intervals, comparable to the use of ECG and pulse oximeter readings [70]. An application was implemented for an Android phone to acquire photoplethysmographic data to test the ECKE protocol. The Android application used the camera on the phone to measure light intensity. When a finger is placed on the lens, as shown in Figure 6.15, the application can measure the amount of light going through the finger at a particular time.

![Figure 6.15: Using the Smart Phone to Obtain the Heartbeat](image)

Two phones were used, Google Nexus One and HTC Desire. One phone was placed on one hand, and the other phone was placed on the other hand. Both phones started measuring data from the camera at the same time, and the light intensity was sent to the laptop through the serial port. Both phones had similar frame rates of approximately 20
frames per second. For each frame the light intensity was measured and sent to the laptop with a timestamp of when the data was collected.

The light intensity was measured using the Java Android APIs. Android allows an application to obtain the raw byte array of the picture, before Android formats it into a lossy image. Both phones (and cameras) use the NV21 format for the raw images. The NV21 format is a planar YUV format, with a 8–bit Y plane followed by an interleaved U and V plane with a 2x2 subsampling. To measure the light intensity the 8–bit Y samples only needed to be obtained. The pseudo-code used is described below.

```java
if (deltaTime < minimalTime) {
    return false;
} else if (deltaTime < overTime) {
    return deltaLight >= threshold;
} else {
    return deltaLight >= endThreshold;
}
```

The 8–bit Y samples are the first two-thirds of the buffer. After the pseudo-code obtains the size of the 8–bit Y samples, the light intensity is measured by adding the 8–bit Y samples together.

The light intensity for each camera frame is measured and then sent to the laptop for further analysis. By having the light intensity values stored, the algorithm was able to be fine-tuned to calculate the inter-pulse interval. The fine-tuning of the measurements enabled future experiments to have the calculation of the inter-pulse interval to be done on the phone. The algorithm needed to be fast and small so that it can run on any smart phone. The pseudo-code to find the inter-pulse interval is described below.

```java
if (deltaTime < minimalTime) {
    return false;
} else if (deltaTime < overTime) {
    return deltaLight >= threshold;
} else {
    return deltaLight >= endThreshold;
}
```

The `deltaTime` is the time between the last heartbeat and now. If the `deltaTime` is less than the minimal time that a heartbeat can occur, then there has not been a new heartbeat. If the `deltaTime` is less than the average time that a heartbeat can occur, then a normal threshold value is used to detect if the light intensity change that has occurred is significant enough for a heartbeat. If the `deltaTime` is greater than the average time that a heartbeat can occur, then a smaller threshold value is used to indicate that a heartbeat has occurred.
The data was gathering was synchronised by having both of the phones attached to the same laptop with USB cables. The USB cables were used as a serial port, and allowed the application running on the laptop. Having the data going to the same application allowed the time synchronisation, since both phones could start measuring at approximately the same time, and finish measuring at approximately the same time.

6.7.7.1 Android Smart Phone Results

Using the method described above a plot of the peaks and troughs was obtained of each of the pulses that were measured. A visual inspection was then performed for each of the graphs to determine if the algorithm was accurate. An example of a graph is shown in Figure 6.16.

![Graph showing light intensity measurements using the smartphone camera on a finger.](image)

Figure 6.16: Light Intensity Measurements using the Smart Phone Camera on a Finger

An example of a graph where there is considerable noise, mainly from the user moving around, is shown in Figure 6.17. The method described above is able to plot the peaks and troughs even if there is noise.

Using the same algorithm similar graphs were obtained for each participant in this research. It was found that better results were obtained when after finding a peak, a trough was then searched for. Also, there was a consistent difference in light intensity measured from the different phones. It was not verified if this is due to the differences in the phones or from different SpO2 levels in the hands.

After obtaining the inter-pulse intervals at each time there was a heartbeat, the pro-
6.7. VALIDATING INTER-PULSE INTERVALS

Figure 6.17: Light Intensity Measurements using the Smart Phone Camera on a Finger

A proposed security protocol was executed, using the inter-pulse intervals from each hand at a particular time. The number of inter-pulse intervals was dependent on the average heartbeat of each of the participants. It took one minute worth of measurements from both hands at the same time from 12 different individuals. The SEV-based EKE protocol was found able to successfully establish a key 64% of the time. When using the ECKE protocol, the inter-pulse interval matched a total of 80% of the time. The range of Inter-Pulse Intervals was found between the 12 individuals to be 1.7 seconds and 0.4 seconds.

6.7.7.2 Arduino Results

A more typical scenario is to validate the proposed protocol using a smart phone and a health sensor. As an initial experiment an off the shelf Arduino pulse sensor [133] was used, which also can supply the inter-pulse interval. Another advantage of using Arduino is that for future experiments wireless communication between the pulse sensor and the smart phone can be easily added. The Arduino device measures the heart rate using an optical heart-rate sensor. The sensor was attached to the finger tip of the subject, while the signal was sent down to the Arduino device.

The sensor outputs a voltage in the range 0 to 5 V, which is normally fed into an ADC of an Arduino Uno; a microcontroller based the ATmega328. Code which runs on the microcontroller is used to calculate the inter-pulse interval and the heart rate. The quantisation of the light intensity using the 10–bit ADC over the 0 to 5 V, causes a step function output, as shown in Figure 6.18.
Analogue to Digital Conversion

Figure 6.18: Light Intensity Measurements using the Arduino Device on a Finger

The problem with using a 10–bit ADC over the 0 to 5 V range, is that it becomes difficult to accurately calculate the peaks and troughs of the wave. Unlike with the phone where the measurements can be quite accurate, the conversion used in the Arduino device is designed more for finding the average heart rate. Obtaining an average heart rate does not require an accurate maximum and minimum detection mechanism.

To get better results the Arduino device ADC was bypassed and the raw data was sent directly to a picoscope, as shown in Figure 6.19. For a more detailed analysis of the setup data was collected using a picoscope 2240 oscilloscope connected to sensor and a laptop. A sampling rate of 1.637 kS/s was used. The experimental setup to gather voltage data from the light intensity sensor placed on a finger is shown in Figure 6.19. The picoscope can obtain the raw voltage data before it gets converted by the ADC found on the Arduino device.

The Arduino device was powered via the USB connection with the laptop. A bread board was used to connect the sensor with the sensor, the picoscope and the Arduino device. The data from the picoscope is transferred to the laptop where it is stored and can be analysed further. The R programming language with the signal package installed is used for further analysis. A ten second interval of the data was examined, and a high frequency noise was removed.

The voltage output of the light intensity sensor using a picoscope is shown in Figure 6.20. There is a dramatic improvement with the ability to detect peaks and troughs. If a 10–bit ADC is used, and then the voltage range should be shifted towards the range for
light intensity going through a finger. The range from 3.5 to 4.0 V is a reasonable value. The plot shows that there is a gradual increase to the peak before there is a sharp decline to the trough. The structure of the light intensity wave is regular with each beat. However, there are noticeable differences in the rate of change and the location of bumps in the wave.

The more accurate that this device can become, the more likely that this device can be used to calculate the most likely value that a less accurate device will be gathering. Having a highly accurate device in the protocol can allow for a more efficient protocol with less communication costs. Another advantage of a highly accurate device is that it can be used as a mechanism to calibrate the less accurate device. The accuracy of a device is increased with the sampling rate as well as the range of values that can be measured. The camera on the phone has a large range of values that can be calculated, however, its sampling rate is dependent on the frame rate of the camera. A low-cost device may have the opposite problem, where the sampling rate may be high, but the range of values can be small.

**6.8 Conclusions**

This first part of this chapter and its contributions have mainly tested the Hypothesis 3, which also helps to answer the Research Question 3. A mobile health care system and
key initialisation mechanisms are examined in detail in this chapter. A description of each component showed the complexity of the system. This chapter showed how different protocols can be used in each component of the mobile health care system. This chapter also demonstrated how physiological data can be used to establish keys between body sensors and other components, where the sensors have no other shared prior secret. GDM is used to effectively extract the requirements of the health care system. The requirements of the key establishment protocol were placed into a Requirement Behaviour Tree and the protocol assumptions were verified.

This second part of this chapter and its contributions have mainly tested the Hypothesis 4, which also helps to answer the Research Question 4. A detailed analysis of inter-pulse intervals was carried out; with security implications of using one inter-pulse interval up to eight inter-pulse intervals are supplied. This chapter proposed a new type of key establishment protocols for a body area network. The protocol takes the error margin or the uncertainty of the sensed data into account. The protocol can establish a key between two devices without the need for any form of initialisation or a priori knowledge. A low entropy key establishment protocol that uses a collisionful hash was also proposed for the BSN environment. The security and performance is compared to the other key establishment protocols when using one inter-pulse interval to eight inter-
pulse intervals. This chapter showed how using a collisionful hash can dramatically reduce the message sizes in the key establishment protocol. Previous protocols that used sensed data to establish keys required wired electrode patches to gather ECG data. This chapter showed how photoplethysmographic data obtained from the video camera on a smart phone can effectively be used as the SEV.

The next chapter will summarise the contributions made in this thesis, and describes several areas of future research.
When you reach for the stars you may not quite get one, but you won't come up with a handful of mud either.

Leo Burnett (1891–1971)

7

Conclusions

This final chapter summarises the research contributions provided by this thesis and suggests several areas for future work.

7.1 Contributions

The ageing population and the increase in chronic diseases have placed a considerable financial burden on health care services. The financial burden can be alleviated by using mobile health care systems. Security is of a major importance to the adoption of the system. Chapter 4 explained why the quality of security services in the mobile health care system is a critical requirement. Key establishment protocols, without the assumption of trusting an individual authentication server, are needed in environments where clients cannot trust individual servers. If the security services of the mobile health care system are down, then the system itself may become compromised. It was shown that with multi-server protocols, even if one or more servers become unavailable or untrustworthy, it may still be possible for the sensor nodes to establish a secure session key. Existing multi-server key establishment protocols were examined in a mobile health care system, and found they may not be suitable in the environment, as shown in Section 4.3.

A critical review of single server protocols developed for both traditional and sensor networks was carried out. The performance of single server protocols, including two of the proposed protocols (NEKA — Protocol 4.6, FPKDC — Protocol 4.8), was analysed in detail. This thesis proposed three multiple server protocols (MSP1 — Protocol 4.12,
CHAPTER 7. CONCLUSIONS

MSP2 — *Protocol 4.13*, MSP3 — *Protocol 4.14*) for sensor networks and complex sensor systems containing multi-tiered networks, and provided a detailed analysis and comparison of each of the protocols, as described in Section 4.5. The protocols were designed by incorporating appropriate security mechanisms based upon the attributes of sensor networks. Implementation of the protocols involved either porting libraries or creating new libraries in TinyOS 2.x. The time elapsed, complexity of the code and memory requirements were analysed in detail on Mica2 sensors. It was demonstrated that a symmetric key implementation has advantages over an asymmetric implementation. The proposed protocols have the added benefit of not solely relying on the sensor nodes to generate cryptographically sound pseudo-random numbers, but using information from each of the sensors to generate the new key. It was shown that the proposed protocols were flexible enough for them to be used with almost any cryptographic primitive and in a range of environments. Different scenarios depending on security, performance and availability were mapped to the proposed protocols, and showed which protocols work best for each of the different scenarios, as shown in Table 4.9. The contributions explained in Chapter 4 tested the Hypothesis 1, which also helps to answer the Research Question 1.

Chapter 5 investigated protocols that use low entropy keys to secure devices, specifically devices that can read the physiological data found on the body. The chapter also proposed new key establishment protocols for both two party (*SEVN — Protocol 5.4*) and three party scenarios (*CSPK — Protocol 5.12*, *ENEKA — Protocol 5.13*, *PNEKA — Protocol 5.14*, *NEKAEV — Protocol 5.15*). The proposed protocols combine traits from a two party password-authenticated key exchange protocol, and a symmetric key server-based key establishment protocol. The proposed protocols have the advantage that they do not require traditional encryption to transport the new session key.

Chapter 5 also proposed a secure group key establishment mechanism for a body area network (*SEVGK — Protocol 5.18*). It allows multiple devices to agree on a shared group key, in an authenticated manner, without the need for any form of initialisation or *a priori* knowledge. It was shown that designing a secure group key agreement protocol is not a trivial task, and discussed different types of attacks. The proposed group protocol use environmental data to achieve authentication. The protocols were implemented in TinyOS and run on Mica2 motes. The time elapsed and memory usage showed that the protocol meets the general BSN requirements. It was demonstrated that a group key establishment protocol can be efficiently run in a sensor network with the RSA algorithm.

An implementation of salient features of the protocols was described and a comparison of the energy consumption of the nodes was supplied, as described in Section 5.9. It was found that when using ECC the communication cost was nearly doubled; the protocols that could be converted to use ECC required larger prime numbers, and were not as well suited for low-resource environments. Also, when mapping data to EC points caused the protocol to become non-deterministic in energy costs. There was a security gain in
using password protocols that can be converted to ECC, however, the security advantages were not significant if the values used in RSA were chosen wisely. The extra requirements needed to enable ECC caused these protocols to be prohibitive. The impact on memory by adding elliptic curves to a sensor application was analysed, revealing that there were additional costs associated with an ECC solution over an RSA solution. The contributions explained in Chapter 5 tested the Hypothesis 2, which also helps to answer the Research Question 2.

Chapter 6 validates the security assumptions of protocols and validates that low entropy data can be generated from low-cost devices. A mobile health care system and key initialisation mechanisms were examined in detail. A description of each component showed the complexity of the system. This thesis showed how different protocols can be used in each component of the health care system. A demonstration of how physiological data can be used to establish keys between body sensors and other components, where the sensors have no other shared prior secret, was also supplied. GDM was used to effectively extract the requirements of the health care system. The requirements of the key establishment protocol were placed into a Requirement Behaviour Tree and the protocol assumptions were verified. The contributions explained in Chapter 6 tested the Hypothesis 3, which also helps to answer the Research Question 3.

A new type of key establishment protocols was proposed for a body area network (ECKE — Protocol 6.5). The protocol takes the error margin or the uncertainty of the sensed data into account. The protocol can establish a key between two devices without the need for any form of initialisation or a priori knowledge. A detailed analysis of inter-pulse intervals was carried out; as well as an analysis of the security implications of using one inter-pulse interval up to eight inter-pulse intervals was supplied. A low entropy key establishment protocol that uses a collisionful hash was also proposed for the BSN environment (ECCH — Protocol 6.7). The security and performance was compared to the other key establishment protocols when using one inter-pulse interval to eight inter-pulse intervals. It was shown that by using a collisionful hash function it can dramatically reduce the message sizes in the key establishment protocol. Previous protocols that used sensed data to establish keys required wired electrode patches to gather ECG data. This study also showed how photoplethysmographic data obtained from the video camera on a smart phone can be used as the low entropy shared secret. The contributions explained in Chapter 6 tested the Hypothesis 4, which also helps to answer the Research Question 4.

7.2 Limitations

As the technology and society evolve, the solutions described in this thesis may make an impact on the success of a mobile health care system. However, there are still some limitations that exist in the contributions described in this thesis.
The protocols have not been tested in a complete mobile health care system. The reliability and low entropy key protocols were tested against a simplified version of the mobile health care system. Also, all the requirements of a mobile health care system had not been added into the GDM design tool. The assumptions made about the simplified mobile health care system may be incorrect. Confirmation that the protocols can still be verified in a complete system is still unknown.

It is assumed that the physiological measurements can be measured with a higher level accuracy, making the new protocols more efficient. If the physiological data can be measured with less uncertainty, then there should be less entropy. However, this hypothesis has not been tested.

The measurements that were used to validate the physiological values was obtained from 54 people over a 24 hour period. Obtaining more data from a single person and obtaining data from more people will increase the reliability of the results. Also, obtaining data from people with heart conditions is important to confirm that the protocols can also work with this set of people.

### 7.3 Future Research

The work presented in this thesis could be extended in a number of directions.

**Integrated system using mobile phones and sensors.** This thesis often circumvented the problem of not having a fully working system by implementing and testing the salient features of the system. Building the entire complex system should also include using Genetic Design Methodology to design the system and the interactions of the component. By creating this design, further validation can occur over the protocol assumption verification in more scenarios. In this thesis, a prototype using Mica2 motes was used to test the performance of the protocols. Also, an Arduino device and Android smart phones were used to prototype a photoplethysmographic sensor to measure IPI physiological values. However, the prototype should be extended to also include an ability to communicate with other sensors and the mobile phone. Also, an ECG sensor should also be developed, and results obtained using ECG and photoplethysmographic should be examined. A fully developed and integrated system will lead to other research areas, such as synchronising the measurements from the different devices, and measuring the amount of energy that is needed by a fully integrated system. The insights brought by the analysis of an integrated and developed system should shed light on limitations and enhancements required in the new security protocols.

**Accurate sensor measurements.** The newly proposed protocols have the potential of strengthening the security of communications by reading the physiological data
found in the body. More accurate measurements of physiological data are an interesting research direction, and an important part of enhancing the security of the new protocols. Another research area that may open up, by obtaining more accurate measurements, is finding other features (other than Inter-Pulse Interval), which can be used to gather more entropy. Also, it is important to find how accurate the instruments can become before noise becomes too large of a factor. Using different types of sensors such as the infra-red sensor found on some smart phones, may be used to measure the pulse in the body, the same way that the camera was used to measure the pulse. Investigating the security relationship between measurements from a more accurate sensor and a less accurate sensor, could further increase the performance of the security protocols.

**Investigate more ECG and IPI data from more people.** The analysis of the ECG data was taken from the PhysioBank database. For this thesis, only a subset of the signals from the database was picked. The PhysioNet database has hundreds more ECG data that should be examined. A subset of 54 healthy people was used since it was assumed that this is more realistic as healthy people should have less randomness than people with heart conditions. Whether or not that assumption is correct, and having an understanding of the distributions from a larger group of people is an important research area for the new protocols. Applying an analysis on the remaining data set will supply us with insight into the randomness of IPI values found in unhealthy people, and indicate if the protocols will have a higher security in unhealthy people compared to healthy people.

**Investigate the physical layer security.** Recent research has investigated using the communication channel for information security. Two devices can exchange a secret over a public wireless fading channel. Unlike conventional secure key establishment methods, this technique uses information theoretic security. The information security is provided because a component of the reciprocal channel fading over time between the two devices is statistically independent with the channel fading from either device to the eavesdropper [24]. However, in some channels such as in BSNs there is not enough randomness in the channels to utilise this interesting area of research [76]. Applying the new protocols into this important area of research could make physical layer security more accessible.
APPENDIX A

ETHICS QUESTIONS

This is a list of questions which should be answered in the negative, before starting experiments on a group of participants.

1. Is it possible for third parties to identify participants, and should this identification be characterised as significant?
2. Does the research involve the participation of people who legally cannot provide voluntary and informed consent for their participation in research?
3. Does the research involve the participation of minors, other than an activity which is highly consistent with standard educational practice?
4. Are the potential participants in an unequal relationship that is likely to impact, or could be perceived to impact, upon either the recruitment process, or the risks associated with the research?
5. Does the research involve the intentional recruitment of Indigenous persons, a significant coincidental recruitment of Indigenous persons, and / or issues likely to be considered significant to Indigenous people?
6. Does the research involve the intentional recruitment of members of a collectivity, a significant coincidental recruitment of members of a collectivity, and / or issues likely to be considered significant to the collectivity?
7. Are drugs, narcotics, poisons, placebo to be ingested / injected, or are invasive procedures to be administered?
8. Does the research involve tissue, blood or other body fluid collection / extraction?
9. Does the research involve a risk of physical injury?
10. Does the research involve exposure to disease or infection?
11. Does the research involve pain or significant discomfort?
12. Does the research involve human exposure to ionising radiation / X-Ray?
13. Does the research involve psychological or emotional stress?
APPENDIX A. ETHICS QUESTIONS

14. Could the research expose participants to potential civil, criminal or other proceedings?
15. Does the research involve sensitive personal information?
16. Could the research expose participants to potential loss of professional reputation, market standing, or employability?
17. Could the research result in significant negative impact upon personal relations?
18. Will potential participants be offered inducements which could be considered coercive?
19. Does the research involve covert observation?
20. Does the research involve deception?
21. Does IS42 or the Commonwealth Privacy Act apply to the research (e.g., access to identified personal data held by third parties subject to privacy regimes)?
22. Does the research involve genetic testing, work with human embryos or foetuses, or other testing which would determine paternity or predilection for a significant medical condition?
23. Does the research require approval as a clinical trial under the CTN or CTX schemes?
24. Will the research be conducted in an overseas setting which is politically unstable and / or where perceived criticism of the government or institutions could attract punitive action?
APPENDIX B

INFORMATION SHEET

This is the information sheet given to each of the participants before running the experiments.

<table>
<thead>
<tr>
<th>Chief Investigator</th>
<th>Student Investigator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vallipuram Muthukkumarasamy</td>
<td>Kalvinder Singh</td>
</tr>
<tr>
<td>School of ICT</td>
<td>School of ICT</td>
</tr>
<tr>
<td>+61755528256</td>
<td>+61755528256</td>
</tr>
<tr>
<td><a href="mailto:v.muthu@griffith.edu.au">v.muthu@griffith.edu.au</a></td>
<td><a href="mailto:Kalvinder.singh@griffithuni.edu.au">Kalvinder.singh@griffithuni.edu.au</a></td>
</tr>
<tr>
<td></td>
<td>Doctor of Philosophy</td>
</tr>
</tbody>
</table>

Why is the research being conducted?

We are investigating the variability of physiological values from the body, to see if it is suitable when establishing secure communication between body sensors. The ageing population and the increase of chronic diseases have placed an immense financial burden on health services. A home health care system, with both body and external sensors may be used to help reduce the costs. Sensors can be used to remotely monitor elderly patients suffering from chronic diseases and allow them to have relatively independent lives. The security or information assurance is a very important criterion of the home health care system. Our research is investigating using physiological values obtained by body sensors to help create secure communication between devices. Finding suitable physiological signal is an ongoing research area. The physiological signals should be the same when measured on different parts of the same body at the same time. An example of a physiological signal that can be measured and that has the above characteristics is heart-rate. We are investigating if blood pressure, oxygen levels and ECG values also have similar characteristics. Griffith University is funding this research, as well as supplying the equipment and personnel to conduct this important work. The research is part of the students study for their doctorate degree.
APPENDIX B. INFORMATION SHEET

What you will be asked to do

Could we have five to ten minutes of your time so that we can take your oxygen levels, pulse, blood pressure, and ECG? We will place a pulse oximeter on your finger to measure your oxygen level and pulse. The blood pressure is measured using Blood Pressure Monitor (Model PG-800A) and will be placed on your wrist. The device will measure your blood pressure and pulse. The ECG is an optional component in our testing, where three leads will be placed on your chest.

The basis by which participants will be selected or screened

Family, friends and workmates will be selected. The participants will be asked if they are over 18 years old. The potential participants will be asked and informed about the research using word of mouth.

The expected benefits of the research

Security is very important in home health care systems, where many people have reservations of installing systems in their home, because of privacy and security concerns. This research will show if physiological values can be used as a basis for establishing secure communication between devices.

Risks to you

The sensors used will be placed on your finger and on your wrist. The devices are commercial instruments, and are FDA approved.

Your confidentiality

No identifiable data will be collected about the people involved in this study.

Your participation is voluntary

Your participation is entirely voluntary. You are free to withdraw from the study at any time. The decision to participate or not to participate does not impact your relationship with Griffith University or the organisation you are in.
Questions / further information

If you have any questions, please contact the chief investigator. His details are at the top of this leaflet.

The ethical conduct of this research

Griffith University conducts research in accordance with the National Statement on Ethical Conduct in Human Research. If you have any concerns or complaints about the ethical conduct of the research project they should contact the Manager, Research Ethics on 3735 5585 or research-ethics@griffith.edu.au.

Feedback to you

This section of the information sheet should outline whether there will be any reporting back of the potential participant’s own results (if appropriate / possible) and any reporting (in an appropriate form) of the overall findings and results of the research.

Privacy Statement

The conduct of this research involves the collection, access and/or use of your identified personal information. The information collected is confidential and will not be disclosed to third parties without your consent, except to meet government, legal or other regulatory authority requirements. A de-identified copy of this data may be used for other research purposes. However, your anonymity will at all times be safeguarded. For further information consult the University’s Privacy Plan at HTTP://WWW.GRIFFITH.EDU.AU/ABOUT-GRIFFITH/PLANS-PUBLICATIONS/GRIFFITH-UNIVERSITY-PRIVACY-PLAN or telephone (07) 3735 5585.
APPENDIX C
CONSENT FORM

<table>
<thead>
<tr>
<th>Chief Investigator</th>
<th>Student Investigator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vallipuram Muthukkumarasamy</td>
<td>Kalvinder Singh</td>
</tr>
<tr>
<td>School of ICT</td>
<td>School of ICT</td>
</tr>
<tr>
<td>+61755528256</td>
<td>+61755528256</td>
</tr>
<tr>
<td><a href="mailto:v.muthu@griffith.edu.au">v.muthu@griffith.edu.au</a></td>
<td><a href="mailto:Kalvinder.singh@griffithuni.edu.au">Kalvinder.singh@griffithuni.edu.au</a></td>
</tr>
</tbody>
</table>

By signing below, I confirm that I have read and understood the information package and in particular have noted that:

- I understand that my involvement in this research will include (include a short summary of what their participation will involve — e.g. the completion of a set of four cognitive tests, on a weekly basis, for three weeks);
- I have had any questions answered to my satisfaction;
- I understand the risks involved;
- I understand that there will be no direct benefit to me from my participation in this research (this may need to be modified for some projects);
- I understand that my participation in this research is voluntary (depending upon the circumstances, there might also be a reference to their decision in no way impacting upon the service they receive from X or their grades);
- I understand that if I have any additional questions I can contact the research team;
- I understand that I am free to withdraw at any time, without comment or penalty;
- I understand that I can contact the Manager, Research Ethics, at Griffith University Human Research Ethics Committee on 3735 5585 (or research-ethics@griffith.edu.au) if I have any concerns about the ethical conduct of the project; and
- I agree to participate in the project.

Name
Signature
Date
# Summary of Notation

Notation is described in each chapter it is used. In this list the main notational conventions are summarised.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$ and $B$</td>
<td>The two nodes who wish to share a new session key.</td>
</tr>
<tr>
<td>$S$</td>
<td>A trusted server.</td>
</tr>
<tr>
<td>$S_i$</td>
<td>A server in a set of servers $S_1, \ldots, S_n$, where $n$ is the numbers of servers.</td>
</tr>
<tr>
<td>$N_A$</td>
<td>A nonce generated by $A$.</td>
</tr>
<tr>
<td>${M}_K$</td>
<td>Encryption of message $M$ with key $K$ to provide confidentiality and integrity.</td>
</tr>
<tr>
<td>$[[M]]_K$</td>
<td>Encryption of message $M$ with key $K$ to provide confidentiality.</td>
</tr>
<tr>
<td>$[M]_K$</td>
<td>One-way transformation of message $M$ with key $K$ to provide integrity.</td>
</tr>
<tr>
<td>$K_{AB}$</td>
<td>The long-term key initially shared by $A$ and $B$.</td>
</tr>
<tr>
<td>$K'_{AB}$</td>
<td>The value of the new session key.</td>
</tr>
<tr>
<td>$K_{AS}, K_{BS}$</td>
<td>Long-term keys initially shared by $A$ and $S$, and by $B$ and $S$ for centralised authentication server.</td>
</tr>
<tr>
<td>$K_{AS_i}, K_{BS_i}$</td>
<td>Long-term keys initially shared by $A$ and $S_i$, and by $B$ and $S_i$, for each $i \in 1, \ldots, n$.</td>
</tr>
<tr>
<td>$X, Y$</td>
<td>The result of the concatenation of data strings $X$ and $Y$.</td>
</tr>
<tr>
<td>$A \rightarrow B : m$</td>
<td>Denotes that $A$ sends a message $m$ to $B$.</td>
</tr>
<tr>
<td>$A \rightarrow S_i : m$</td>
<td>Denotes that $A$ sends a message $m$ to each server.</td>
</tr>
<tr>
<td>$S_i \rightarrow A : m$</td>
<td>Denotes that each server sends a message $m$ to $A$.</td>
</tr>
<tr>
<td>$m$</td>
<td>Another way to define sending of message $m$.</td>
</tr>
<tr>
<td>$X \oplus Y$</td>
<td>Exclusive-or operation with $X$ and $Y$.</td>
</tr>
<tr>
<td>$\Upsilon$</td>
<td>Cryptographically strong key generated from the environment.</td>
</tr>
<tr>
<td>$\pi$</td>
<td>Non–changing low entropy key (e.g., a password).</td>
</tr>
<tr>
<td>$\Psi$</td>
<td>Changing low entropy key (e.g., Inter–Pulse Interval of the heart).</td>
</tr>
</tbody>
</table>
## SUMMARY OF ACRONYMS

The acronyms are described in each chapter it is used. In this list the main acronyms are summarised.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3PKDP</td>
<td>Three Party Key Distribution Protocol</td>
</tr>
<tr>
<td>$\mu$TESLA</td>
<td>Microtimed Efficient Stream Loss-Tolerant Authentication</td>
</tr>
<tr>
<td>aGPS</td>
<td>Assisted Global Positioning System</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BSN</td>
<td>Body Sensor Network</td>
</tr>
<tr>
<td>bpm</td>
<td>beats per minute</td>
</tr>
<tr>
<td>CA</td>
<td>Certificate Authority</td>
</tr>
<tr>
<td>CISSP</td>
<td>Certified Information Systems Security Professional</td>
</tr>
<tr>
<td>CSPK</td>
<td>Combined Symmetric Password Key</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CMVP</td>
<td>Cryptographic Module Validation Program</td>
</tr>
<tr>
<td>DBT</td>
<td>Design Behaviour Tree</td>
</tr>
<tr>
<td>ECG</td>
<td>Elliptic Curve Cryptography</td>
</tr>
<tr>
<td>ECCH</td>
<td>Error Correcting Collisionful Hash</td>
</tr>
<tr>
<td>ECG</td>
<td>Electrocardiography</td>
</tr>
<tr>
<td>ECKE</td>
<td>Error Correcting Key Establishment</td>
</tr>
<tr>
<td>EKG</td>
<td>Electrocardiogram</td>
</tr>
<tr>
<td>EKE</td>
<td>Encrypted Key Exchange</td>
</tr>
<tr>
<td>ENEKA</td>
<td>Enhanced No Encryption Key Agreement</td>
</tr>
<tr>
<td>FPKDC</td>
<td>Five-Pass Key Distribution Centre</td>
</tr>
<tr>
<td>GDM</td>
<td>Genetic Design Methodology</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HD</td>
<td>High Definition</td>
</tr>
<tr>
<td>HHC</td>
<td>Home Health Controller</td>
</tr>
<tr>
<td>HMAC</td>
<td>Hash-based Message Authentication Code</td>
</tr>
<tr>
<td>HRV</td>
<td>Heart Rate Variance</td>
</tr>
<tr>
<td>HSN</td>
<td>Home Sensor Network</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>IBT</td>
<td>Integrated Behaviour Tree</td>
</tr>
<tr>
<td>IPI</td>
<td>Inter-Pulse Interval</td>
</tr>
<tr>
<td>IQD</td>
<td>InterQuartile Distance</td>
</tr>
<tr>
<td>KDC</td>
<td>Key Distribution Centre</td>
</tr>
<tr>
<td>MAC</td>
<td>Message Authentication Code</td>
</tr>
<tr>
<td>MCU</td>
<td>Multipoint Control Unit</td>
</tr>
<tr>
<td>MSP</td>
<td>Multiple Server Protocol</td>
</tr>
<tr>
<td>NEKA</td>
<td>No Encryption Key Agreement</td>
</tr>
<tr>
<td>NEKAEV</td>
<td>No Encryption Key Agreement with Environment Values</td>
</tr>
<tr>
<td>OHC</td>
<td>One-way Hash Chain</td>
</tr>
<tr>
<td>PAK</td>
<td>Password Authenticated Key exchange</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>PIN</td>
<td>Personal Identification Number</td>
</tr>
<tr>
<td>PIRI</td>
<td>Plan Implement Review Improve</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>PPG</td>
<td>Photoplethysmogram</td>
</tr>
<tr>
<td>PPK</td>
<td>Password Protected Key exchange</td>
</tr>
<tr>
<td>PNEKA</td>
<td>Password No Encryption Key Agreement</td>
</tr>
<tr>
<td>PRF</td>
<td>PseudoRandom Function</td>
</tr>
<tr>
<td>PSKA</td>
<td>Physiological-Signal-based Key Agreement</td>
</tr>
<tr>
<td>QoP</td>
<td>Quality of Privacy</td>
</tr>
<tr>
<td>RA</td>
<td>Registration Authority</td>
</tr>
<tr>
<td>RBT</td>
<td>Requirement Behaviour Tree</td>
</tr>
<tr>
<td>RSA</td>
<td>Ron Rivest, Adi Shamir and Leonard Adleman</td>
</tr>
<tr>
<td>SEV</td>
<td>Secure Environment Value</td>
</tr>
<tr>
<td>SEVGK</td>
<td>Secure Environment Value Group Key</td>
</tr>
<tr>
<td>SEVN</td>
<td>Secure Environment Values for Nonces</td>
</tr>
<tr>
<td>SPEKE</td>
<td>Secure Password Exponential Key Exchange</td>
</tr>
<tr>
<td>SPINS</td>
<td>Security Protocols for Sensor Networks</td>
</tr>
<tr>
<td>SpO2</td>
<td>Oxygen Saturation</td>
</tr>
<tr>
<td>SQRT</td>
<td>Square Root</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language</td>
</tr>
<tr>
<td>UWB</td>
<td>Ultra-wideband</td>
</tr>
<tr>
<td>VA</td>
<td>Validation Authority</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
</tbody>
</table>
List of References


LIST OF REFERENCES


[132] Bartosz Przydatek, Dawn Song and Adrian Perrig. ‘SIA: secure information aggregation in sensor networks’. In: Proceedings of the 1st international conference on Embedded networked sensor systems. SenSys ’03. Los Angeles, California, USA:


