Western Graduate&PostdoctoralStudies

Western University
Scholarship@Western

Electronic Thesis and Dissertation Repository

6-24-2024 11:00 AM

A Secure Lightweight Wireless M-Bus Protocol for IoT: Leveraging the Noise Protocol Framework

Wafaa Anani,

Supervisor: Dr. Abdelkader Ouda, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Electrical and Computer Engineering
© Wafaa Anani 2024

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Part of the Digital Communications and Networking Commons, and the Electrical and Computer Engineering Commons

Recommended Citation

Anani, Wafaa, "A Secure Lightweight Wireless M-Bus Protocol for IoT: Leveraging the Noise Protocol Framework" (2024). *Electronic Thesis and Dissertation Repository*. 10181.
https://ir.lib.uwo.ca/etd/10181

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlswadmin@uwo.ca.

Abstract

The expansion of smart metering within the Internet of Things (IoT) ecosystem underscores the need for robust security protocols that safeguard data transmission while optimizing device efficiency. Wireless Meter-Bus (wM-Bus), a key protocol for remote meter reading in utility systems such as gas, water, and heat meters, faces significant security challenges. This dissertation introduces a method to enhance wM-Bus security by integrating the Noise Protocol Framework (NPF), which secures wM-Bus against vulnerabilities and optimizes for the energy constraints of IoT devices. Initially examining wM-Bus security issues, particularly in battery-operated smart meters, the study explores the NPF's lightweight, adaptable security solutions. Implementation analysis focuses on NPF handshake patterns NX (non-interactive with public key transmission by the initiator) and XX (mutual public key exchange), assessing their compatibility with wM-Bus through metrics such as memory use, packet size, and handshake time. Findings reveal that these patterns significantly outperform traditional methods like Transport Layer Security (TLS) in reducing energy consumption, thereby extending IoT devices' operational lifespan. The study achieved a 5% battery-life reduction with NX and a 25% battery-life reduction with XX, enhancing both security and efficiency. These implementations also improved system security by reducing handshake times by up to 4.7% and minimizing packet sizes by up to 68.38%, critical for mitigating security threats. They also showed improvement in memory consumption compared to TLS. The proposed lightweight protocol effectively balances advanced security and efficiency, maintaining data confidentiality, integrity, and availability in smart metering without sacrificing performance. Security testing against the Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege (STRIDE) model confirmed the resilience of this new protocol, thereby enhancing the security framework. This research not only establishes a more secure foundation for smart metering but also sets a precedent for future studies on integrating lightweight cryptographic frameworks in IoT environments.

Keywords Security, Wireless communication, Wireless M-Bus, Noise Protocol Framework, Internet of Things, Protocols, Batteries, LPWAN, Smart Metering, Ecosystem

Summary for Lay Audience

The dissertation focuses on improving security for devices like smart meters that communicate wirelessly, especially within the ”Internet of Things” (IoT) framework. These devices, which measure things like water, gas, and electricity usage, use a communication method known as Wireless M-Bus. However, this method has security risks. The dissertation introduces a new way to make these communications safer using something called the Noise protocol, which is better suited for devices with limited battery life, like those in IoT setups. The study shows that this new method reduces the battery usage of these devices and enhances security without sacrificing performance, making it a practical solution for ensuring that data transmitted by smart meters is both secure and efficient.

Epigraph

”Amidst the trials, my spirit never wavered; in the shadow of doubt, hope remained my beacon. This work is a tribute to that enduring light, to my homeland Palestine”

— W. Anani

Dedication

To The memory of my grandparents will always be cherished, as they instilled wisdom, love, and unwavering support throughout my life. Their laughter, comforting embraces, and selfless acts of kindness have left a lasting impression on my heart and their memory serves as a constant source of inspiration for me to strive for personal growth. Their love and guidance will always be remembered and treasured, and they will forever hold a special place in my heart.

Acknowledgements

I would like to express my sincere gratitude to the following people who have made my PhD journey a memorable and fulfilling experience.

First and foremost, I would like to thank God for providing me with the strength, courage, and perseverance to complete my PhD program. Next, I would like to express my heartfelt appreciation to my parents for their constant encouragement and support. They have been a source of inspiration throughout my journey and I am grateful for their love and support. Also, my sisters and brothers who have always been there for me.

I would also like to express my immense gratitude to my supervisor, Dr. Abdelkader Ouda, for his tremendous support and guidance throughout my PhD journey. His unwavering expertise and support have been instrumental in shaping my research, and I cannot express enough how grateful I am for all he has done for me. I truly appreciate the opportunities he has provided me and the time and effort he has dedicated to my success. I am also grateful for the support and encouragement from my classmates (Aisha Edrah, Iman Abu-Sulyam, Yazan Aref) and all of my friends who believed in me and motivated me to keep going.

I would also like to express my gratitude to my Advisory Committee for their insights and guidance throughout my PhD program. Their knowledge and experience have been invaluable, and I appreciate the time and effort they have dedicated to my research.

Finally, I would like to thank my wellness consultant, Sara Hanna, who helped me during my toughest times. Your guidance and support have been a source of strength, and I am grateful for your help.

Thank you all for being a part of my journey.

W. Anani

Table of Contents

Abstract **ii**

Summary for Lay Audience **iv**

Epigraph **v**

Dedication **vi**

Acknowledgements **vii**

Table of Contents **viii**

List of Tables **xiv**

List of Figures **xvii**

List of Abbreviations **xxiv**

1 Introduction **1**

1.1 Overview 1

1.2 Motivation 1

1.3 Research Goal and Objectives 2

1.4 Research Contribution 3

1.5 Research Methodology 6

1.5.1 Introduction 6

1.5.2 PART I: Wireless Communication Technologies Investigation . . . 7

1.5.3 PART II: Research Design - Framework Implementation 8

1.5.4 Data Collection and Tools 10

1.5.5 Variables Measures and Evaluation 11

1.5.6 Consideration and Limitation 12

1.6 Thesis Organization 13

2 Background and Literature Review 14

2.1 Background 14
  2.1.1 Overview of Smart Grid and IoT Technologies 14
  2.1.2 Importance of Communication Technologies 15
  2.1.3 wM-Bus Protocol 15
  2.1.4 Security Challenges 17
  2.1.5 NPF 17
  2.1.6 Integration of wM-Bus with NPF 20
2.2 Literature Review 21
  2.2.1 Security Aspects of Smart Meter Infrastructures 21
  2.2.2 wM-Bus Implementation & Security in Smart Metering 23
  2.2.3 NPF and its Applications 24

3 Demystifying Wireless Technologies for Best Uses in IoT Echo-Systems 26

3.1 Overview 26
3.2 Wireless Technologies 28
  3.2.1 Short Range - WPAN and WHAN 28
    Bluetooth and BLE 28
    ZigBee and ZigBee PRO 28
    Z-Wave/Z-Wave Plus 29
  3.2.2 Medium Range - WLAN 31
    Wi-Fi 31
    Wi-Fi Halow 31
    Wi-SUN 32
    WM-Bus 32
    DASH7 33

3.2.3 Long Range - WWAN 35
LoRa 35
Sigfox 35
LTE-M 36
NB-IoT 36
RPMA 37
3.3 Wireless Technologies For Echo-System Applications 39
3.3.1 IoT Echo-System Requirements 39
3.3.2 Wireless Technologies for Smart Metering 40
3.4 Wireless Meter Bus: Secure Remote Metering within the IoT Smart Grid . . 42
3.4.1 Overview 42
3.4.2 wM-Bus 44
wM-Bus Overview 44
wM-Bus Protocol Stack 44
wM-Bus Communication Modes 45
wM-Bus Data/Message Frame 47
wM-Bus Smart Grid Infrastructure 47
wM-Bus Security 48
3.4.3 Proposed wM-Bus Security Framework 49
Overall Architecture 49
3.4.4 Security Framework base configuration 50
The Proposed Security Framework Aspects 50
Summary 52

4 Lightweight Protocol Over wM-Bus for IoT Echosystems 54
4.1 Protocol Overview 54
4.2 Protocol Communications 54
4.3 Protocol Assumptions 57

4.4 Protocol Design Methodology 57
4.5 Protocol Implementation Setup 58

5 Implementation of Protocols to Secure Wireless Communication 61
5.1 Introduction 61
5.2 Phase One: NPF 66
5.2.1 NPF Predefined Pattern Analysis 66
XN Pattern 67
KK Pattern 67
NN Pattern 68
IN Pattern 69
XK Pattern 70
NK Pattern 71
IK / KX / XX Patterns Pattern 71
NX Pattern 74
IX Pattern 74
XN / KN Pattern 75
5.2.2 NPF Implementation 77
5.3 Phase Two: wM-Bus Protocol 82
5.3.1 Hardware Configuration 82
5.3.2 Software Configuration 84
RCTools 84
5.3.3 wM-Bus Development Implementation 86
wM-Bus Gateway Implementation 87
wM-Bus Meter Implementation 88
5.4 Phase Three: wM-Bus with TLS 1.2 94
5.4.1 wM-Bus with TLS - Meter Analysis 96
5.4.2 wM-Bus with TLS - Gateway Analysis 99

5.4.3 wM-Bus with TLS - Handshake Meter & Gateway Combined Analysis . . . 103
5.5 Phase Four: wM-Bus with NPF . . . 108
5.5.1 wM-Bus with NPF - Handshake Gateway . . . 109
5.5.2 wM-Bus with NPF - Handshake Meter . . . 109
5.5.3 wM-Bus with NPF - Handshake . . . 110
Meter Memory Usage . . . 111
Meter Elapsed Time . . . 111
5.5.4 wM-Bus With NPF Handshake Cumulative Analysis . . . 114
5.6 Phase Five: Optimizing NPF for Lightweight Protocol . . . 121
5.6.1 NPF - Two Messages . . . 123
IK Pattern . . . 123
IN Pattern . . . 125
IX Pattern . . . 134
KK Pattern . . . 134
KN Pattern . . . 140
KX Pattern . . . 141
NK Pattern . . . 147
NN Pattern . . . 150
NX Pattern . . . 154
5.6.2 NPF - Three Messages . . . 157
XK Pattern . . . 157
XN Pattern . . . 160
XX Pattern . . . 161

6 Evaluation and Analysis 167

6.1 Nosie Protocol Framework Patterns Comparative Analysis . . . 167
6.2 Comparison Between TLS and NPF NX and XX . . . 172

6.2.1 Memory Usage . . . 172
6.2.2 Packet Size . . . 172
6.2.3 Handshake Time . . . 173
6.2.4 Summary . . . 173
6.3 STRIDE Threat Models . . . 174
6.3.1 STRIDE Threat Models Overview . . . 174
6.3.2 Define the Components . . . 175
6.3.3 Threat Model for NX Pattern . . . 176
6.3.4 Threat Model for XX Pattern . . . 177
6.3.5 Noise Explorer . . . 179
6.4 Battery Life Analysis . . . 179
6.4.1 Power Consumption and Battery Life . . . 179
6.4.2 Battery Life Calculation for NX Pattern . . . 181
6.4.3 Battery Life Calculation for XX Pattern . . . 184
6.4.4 Battery Life Calculation for TLS . . . 185

7 Conclusion and Future Work 186
7.1 Summary of Contributions . . . 186
7.2 Research Limitation . . . 190
7.3 Future Research Directions . . . 190

Bibliography 192

Curriculum Vitae 198

List of Tables

3.1 Pros and Cons of Bluetooth and BLE 29

3.2 Pros and Cons of ZigBee 29

3.3 Pros and Cons of Z-Wave 29

3.4 Short Range Wireless Technology Specifications 30

3.5 Pros and Cons of Wi-Fi 31

3.6 Pros and Cons of Wi-Fi HaLow 32

3.7 Pros and Cons of Wi-SUN 32

3.8 Pros and Cons of WM-Bus 33

3.9 Pros and Cons of DASH7 (D7) 33

3.10 Medium Range Wireless Technology Specifications 34

3.11 Pros and Cons of LoRa 35

3.12 Pros and Cons of Sigfox 36

3.13 Pros and Cons of LTE-M 36

3.14 Pros and Cons of NB-IoT 37

3.15 Pros and Cons of RPMA 37

3.16 Long Range Wireless Technology Specifications 38

3.17 Wireless Technology Suitable Uses for IoT application 41

3.18 Wireless M-BUS Transfer Modes 45

3.19 Wireless M-Bus Frame Format A [60] 47

3.20 Wireless M-Bus Frame Format B [60] 48

3.21 Wireless M-Bus Security Standards 49

3.22 Wireless M-Bus Security Configuration 50

5.1 Noise Framework Predefined Patterns 66

5.2 Noise Framework Predefined Patterns – Security Conclusion 78

5.3 Noise Framework presenting the six patterns with the result as in memory usage, time, and packet size 80
5.4 Wireless M-Bus Radiocrafts DK RC1701HP-MBUS4 Specifications 83
5.5 List of Wireless M-Bus fields 87
5.6 List of Wireless M-Bus Message Types 87
5.7 Wireless M-Bus with TLS Handshake Memory Usage and Elapsed Time For the Meter (Client) 97
5.8 Wireless M-Bus with TLS Handshake Memory Usage and Elapsed Time For the Gateway (Server) 101
5.9 Wireless M-Bus with TLS Handshake Memory Usage and Elapsed Time For both Gateway and Meter 105
5.10 Wireless M-Bus with Noise Framework Handshake Calculations – Gateway 109
5.11 Wireless M-Bus with Noise Framework Handshake Calculations – Meter . . 109
5.12 Wireless M-Bus with Noise Framework Handshake between the Meter and Gateway 110
5.13 IK Pattern Handshake - Meter (Client) 125
5.14 IK Pattern Handshake - Gateway (Server) 125
5.15 IN Pattern Handshake - Meter (Client) 132
5.16 IN Pattern Handshake - Gateway (Server) 132
5.17 IX Pattern Handshake - Meter (Client) 135
5.18 IX Pattern Handshake - Gateway (Server) 135
5.19 KK Pattern Handshake - Meter (Client) 138
5.20 KK Pattern Handshake - Gateway (Server) 138
5.21 KN Pattern Handshake - Meter (Client) 141
5.22 KN Pattern Handshake - Gateway (Server) 142
5.23 KX Pattern Handshake - Meter (Client) 144
5.24 KX Pattern Handshake - Gateway (Server) 145

5.25 NK Pattern Handshake - Meter (Client) 148

5.26 NK Pattern Handshake - Gateway (Server) 148

5.27 NN Pattern Handshake - Meter (Client) 151

5.28 NN Pattern Handshake - Gateway (Server) 152

5.29 NX Pattern Handshake - Meter (Client) 155

5.30 NX Pattern Handshake - Gateway (Server) 155

5.31 XK Pattern Handshake - Meter (Client) 158

5.32 XK Pattern Handshake - Gateway (Server) 158

5.33 XN Pattern Handshake - Meter (Client) 161

5.34 XN Pattern Handshake - Gateway (Server) 162

5.35 XX Pattern Handshake - Meter (Client) 164

5.36 XX Pattern Handshake - Gateway (Server) 165

6.1 Comparison of all Noise Framework Patterns 169

6.2 TLS Vs Noise Framework to Secure Wireless M-Bus 174

6.3 STRIDE Threat Models 175

6.4 The handshake NX pattern messages 176

6.5 The handshake XX pattern messages 177

6.6 Battery life (year) 180

List of Figures

2.1 NPF's XX Pattern: outlines the message flow for the XX handshake pattern, indicating the sequential exchange of ephemeral keys (e), static keys (s), and the corresponding operations (ee, es) performed between two parties to achieve mutual authentication and establish a secure communication channel. 19

2.2 Overview of NPF: Handshake patterns and associated cryptographic algorithms . . . 19

3.1 Wireless Technology Ranges vs Data Rate . . . 28

3.2 Wireless Meter Bus (wM-Bus) Protocol Stack . . . 44

3.3 Bidirectional communication (Mode S2) . . . 46

3.4 Example of a wM-Bus packet . . . 47

3.5 wM-Bus: Remote Metering Infrastructure . . . 48

3.6 Example of a wM-Bus message that includes an encryption configuration and other data added by the model [6] . . . 51

3.7 wM-Bus: Remote Metering Infrastructure . . . 52

4.1 Overview of the proposed model: communication setup demonstrating the integration of the wM-Bus protocol with NPF for secure data transmission . 55

4.2 wM-Bus Packet Frame . . . 55

4.3 Secure wM-Bus Communication Handshake Using the NPF with XX Pattern Between Initiator (Meter) and Responder (Gateway) . . . 56

4.4 wM-Bus development kit RC1701HP-MBUS4-DK . . . 59

4.5 A wM-Bus communication setup with a laptop connected to a Gateway (Responder), and a Meter (Initiator), part of a Radiocrafts demonstration kit. 60

5.1 Activity Diagram for Implementation and Evaluation Phases . . . 62

5.2 Sample of a terminal view of the NPF communication for IX Pattern 79

5.3 Total Memory Usage of the Six Patterns for both Client and Server 80
5.4 Total Elapsed Time of the Six Patterns for both Client and Server 81
5.5 Enter Caption 82
5.6 Total Memory Usage of each NPF Pattern 83
5.7 Total Elapsed Time of each NPF Pattern 83
5.8 wM-Bus Development Kit – Software Tool RC232-CCT for Configuration . 84
5.9 wM-Bus Development Kit – Software M-Bus-Demo for Testing 85
5.10 wM-Bus Development Kit – Software M-Bus-Demo for Testing 85
5.11 Example of wM-Bus Packet 86
5.12 Sample of python code using different type of message (CI) with different lengths 87
5.13 Sample of the Meter(Client) log for the calculated memory usage and elapsed time 88
5.14 wM-Bus Memory Usage for different Message Types (7A, 72, 8) with different message length 89
5.15 wM-Bus Time Elapsed for different Message Types (8, 72, 7A) with different message length 89
5.16 wM-Bus Time Elapsed for different Message Types (8, 72, 7A) with different message length 90
5.17 Terminal View – Example of the message “08” with length 75 Bytes. Showing the raw message and the packet header fields for the gateway 90
5.18 wM-Bus meter memory usage for three different messages at different length 91
5.19 wM-Bus meter memory usage for three different messages at different length 92
5.20 wM-Bus meter memory usage for three different messages at different length 92
5.21 wM-Bus meter memory usage for three different messages at different length 93
5.22 wM-Bus Gateway and Meter memory usage for three different messages at different length 93

5.23 wM-Bus Gateway and Meter elapsed time for three different messages at different length 94
5.24 TLS cipher suite used for wM-Bus implementation 95
5.25 One example of a TLS Cipher suite: 'EDCDH-ECDSA-AES128-SHA' . . . 95
5.26 Example of the log file for one Cipher suite at the Meter side. 96
5.27 wM-Bus with TLS shows handshake memory usage for the meter with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges. 98
5.28 wM-Bus with TLS showing handshake memory usage for the meter with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges 99
5.29 wM-Bus with TLS showing handshake elapsed time usage for the meter with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges 100
5.30 wM-Bus with TLS showing memory usage for the meter with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges 100
5.31 wM-Bus with TLS showing handshake Memory Usage for the Gateway with the combination of 8 different cipher suites, 3 different wM-Bus messages and two different key exchanges 102
5.32 wM-Bus with TLS showing memory usage for the Gateway with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges 102
5.33 wM-Bus with TLS showing handshake elapsed time for the Gateway with the combination of 8 different cipher suites, 3 different wM-Bus messages and two different key exchanges 103

5.34 wM-Bus with TLS showing elapsed time for the Gateway with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges 104
5.35 wM-Bus with TLS showing handshake Memory Usage with the combination of 8 different cipher suites, 3 different wM-Bus messages and two different key exchanges 105
5.36 wM-Bus with TLS showing Memory Usage with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges 106
5.37 wM-Bus with TLS showing handshake Elapsed Time with the combination of 8 different cipher suites, 3 different wM-Bus messages and two different key exchanges 106
5.38 wM-Bus with TLS showing Elapsed Time with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges 107
5.39 wM-Bus with TLS showing handshake Elapsed Time with the combination of 8 different cipher suites, 3 different wM-Bus messages and two different key exchanges 107
5.40 wM-Bus with TLS showing Elapsed Time with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges 108
5.41 Memory Usage Handshake Initiate 112
5.42 Memory Usage Handshake ClientHello 112
5.43 Memory Usage Handshake Receive ServerHello 113
5.44 Memory Usage Handshake Send Application Data 113
5.45 Memory Usage Handshake Initiate 115
5.46 Memory Usage Handshake ClientHello 115

5.47 Memory Usage Handshake Receive ServerHello 116
5.48 Memory Usage Handshake Send Application Data 116
5.49 Memory Usage wM-Bus With NPF Handshake for all Handshake steps . . . 117
5.50 Elapsed Time wM-Bus with NPF Handshake for all Handshake steps 118
5.51 wM-Bus with NPF Handshake Total Memory Usage 120
5.52 wM-Bus with NPF Handshake Total Elapsed Time 120
5.53 Sample of the iteration of all patterns with various ciphers, key exchange (DH), and hashing algorithms 122
5.54 Sample of the terminal view during the execution of the patterns with different security options 123
5.55 Meter Memory Usage associated with different security options in the IK pattern 126
5.56 Meter Packet size associated with different security options in the IK pattern 126
5.57 Meter Elapsed Time associated with different security options in the IK pattern 127
5.58 Gateway Memory Usage associated with different security options in the IK pattern 127
5.59 Gateway Packet Size associated with different security options in the IK pattern 128
5.60 Gateway Elapsed Time associated with different security options in the IK pattern 129
5.61 IK Handshake Memory Usage associated with different security options . . 130
5.62 IK Handshake Packet Size associated with different security options 130
5.63 IK Handshake Elapsed Time associated with different security options . . . 131
5.64 IN Handshake Memory Usage associated with different security options . . 132
5.65 IN Handshake Packet Size associated with different security options 133
5.66 IN Handshake Elapsed Time associated with different security options . . . 133

5.67 IX Handshake Memory Usage associated with different security options . . 135

5.68 IX Handshake Packet Size associated with different security options 136

5.69 IX Handshake Elapsed Time associated with different security options . . . 136

5.70 KK Handshake Memory Usage associated with different security options . . 139

5.71 KK Handshake Packet Size associated with different security options 139

5.72 KK Handshake Memory Usage associated with different security options . . 140

5.73 KN Handshake Memory Usage associated with different security options . . 142

5.74 KN Handshake Packet Size associated with different security options 143

5.75 Enter Caption 143

5.76 KX Handshake Memory Usage associated with different security options . . 145

5.77 KX Handshake Packet Size associated with different security options 146

5.78 KX Handshake Elapsed Time associated with different security options . . 146

5.79 NK Handshake Memory Usage associated with different security options . . 149

5.80 NK Handshake Packet Size associated with different security options 149

5.81 NK Handshake Elapsed Time associated with different security options . . 150

5.82 NN Handshake Memory Usage associated with different security options . . 152

5.83 NN Handshake Packet Size associated with different security options 153

5.84 NN Handshake Elalapsed Time associated with different security options . . 153

5.85 NX Handshake Memory Usage associated with different security options . . 155

5.86 NX Handshake Packet Size associated with different security options 156

5.87 NX Handshake Packet Size associated with different security options 156

5.89 Xk Handshake Packet Size associated with different security options 159

5.88 XK Handshake Memory Usage associated with different security options . . 159

5.90 XK Handshake Packet Size associated with different security options 160

5.91 XN Handshake Memory Usage associated with different security options . . 162

5.92 XN Handshake Packet Size associated with different security options 163

5.93 XN Handshake Elapsed Time associated with different security options . . 163

5.94 XX Handshake Memory Usage associated with different security options . . 165

5.95 XX Handshake Packet Size associated with different security options 166

5.96 XX Handshake Elapsed Time associated with different security options . . 166

6.1 Sample of Noise Explorer Verification Tool for XX pattern 180

6.2 Battery life for the different Protocol 181

List of Abbreviations

Abbreviations

AEAD	Authenticated encryption with associated data
AESGCM	AES with Galois/Counter Mode
AES	Advanced Encryption Standard
AFL	Authentication and Fragmentation Layer
AMI	Advanced Metering Infrastructure
APL	Applicaiton Layer
AP	Acess Point
BLE	Bluetooth Low Energy
CIoT	Celluar IoT
DH	Diffie-Hellman
DLL	Data Link Layer
DUC	Data Unit Collector
ECDHE	Elliptic-Curve Diffie-Hellman Ephemeral
ELL	Extended Data Link Layer
FANs	Field Area Networks
HMACs	Hashed Message Authentication Codes

IIoT	Industrial Internet of Things
IoT	Internet of Things
ISM	The Industrial, Scientific and Medical
IWSN	industrial wireless sensors networks
KDF	Key Derivation Function
LPWAN	Low Power Wide Area Networks
M-Bus	Meter Bus
NB-IoT	Narrow-band Internet of Things
NPF	Noise Protocol Framework
OFDM	Orthogonal Frequency Division Modulation
OTAA	Over-The-air-Activation
PHY	Physical Layer
PIN	Personal Identification Number
SHA	Secure Hash Algorithm
STA	Station
TLP	Transport Layer
WEP	wired Equivalent Privacy
WHAN	Wireless Home Area Network

WLAN	Wireless Local Area Network
wM-Bus	Wireless Meter Bus
WNAN	Wireless Neighborhood Area Network
WPAN	Wireless Personal Area Network

Chapter 1

1 Introduction

1.1 Overview

The Internet of Things (IoT) represents a transformative shift in the digital landscape, interconnecting a vast array of devices, from everyday household items to sophisticated industrial tools, through the internet. This emerging network is crucial in contemporary technology ecosystems, utilizing data collection and analysis to improve efficiency, functionality, and user experience across diverse domains. Particularly in utility management, IoT innovations such as smart meters are revolutionizing the way utilities like gas, water, and electricity are monitored and managed, facilitating real-time data tracking, consumption optimization, and predictive maintenance. Central to the IoT's efficacy in these applications is the role of wireless communication technologies, among which Wireless Meter-Bus (wM-Bus) [1] stands out. As a European standard for wireless meter reading, wM-Bus plays a crucial role in enabling the remote collection and transmission of utility usage data. This not only streamlines operational processes but also empowers consumers with immediate insights into their consumption patterns, driving both economic and environmental benefits.

1.2 Motivation

As we explore deeper into the digital integration of utility management through devices like smart meters, the criticality of robust security frameworks becomes increasingly apparent. The wM-Bus protocol, designed for the remote reading of utility meters, stands at the core of this infrastructure, facilitating efficient data transmission across vast networks. However, its current security mechanisms exhibit vulnerabilities that could be exploited to compromise data confidentiality, integrity, and system availability, leading to potential unauthorized access, data manipulation, and disruptions in service. These vulnerabilities not only pose a risk to the privacy and trust of consumers but also threaten the resilience and reliability of smart grid systems.

In addressing the prevailing practices within the industry or utility management, it's evident that security measures are often overlooked in favor of easier deployment and cost reduction. Typically, the focus is on installing meters with minimal initial investment and operational interference, intending not to require maintenance or replacement for a duration of 10 to 12 years. However, when security protocols are implemented, they frequently rely on a uniform key for all meters, usually the default manufacturer's key provided with a batch of meters. This approach poses a significant risk; if a single key is compromised, it renders all meters vulnerable, potentially jeopardizing the entire network.

Acknowledging these challenges, as our initiative seeks to challenge and change these practices by introducing robust, adaptable security solutions that safeguard individual meters and the broader system against such vulnerabilities, thus enhancing the overall security posture of IoT utility management systems. The adoption of the Noise Protocol Framework (NPF) emerges as a beacon of hope. With its potential security properties, NPF presents an opportunity to significantly enhance the security landscape of wM-Bus communications [2]. The objective is to establish a framework of comprehensive security measures that provide protection against existing threats while remaining flexible to address future challenges. This initiative extends past the simple correction of vulnerabilities; it seeks to fundamentally transform the security paradigm to maintain alignment with the rapidly evolving IoT ecosystem.

By integrating the wM-Bus protocol with the advanced cryptographic techniques of the NPF, we aim to establish a NPF benchmark for security within the IoT space. This research is not just a technical improvement; it represents a dedication to ensuring the long-term, reliability, and credibility of smart metering infrastructures globally. As we embark on this journey, our vision is clear: to foster a secure, resilient IoT environment where smart technologies thrive, bolstered by the confidence of consumers and the robustness of our security frameworks. This is the cornerstone upon which the future of IoT utility management will be built, ensuring a seamless blend of innovation and security.

1.3 Research Goal and Objectives

The primary goal of this thesis is to utilize NPF to improve the wM-Bus security protocols by optimizing various factors such as packet size, memory usage, handshake time, and power consumption to achieve strong protection while minimizing the impact on communication efficiency in IoT ecosystems.

The objectives that support this goal include:

- Minimize the impact on packet size to ensure efficient data transmission over the wM-Bus network, aiming to keep any increase in packet size minimal compared to TLS.

- Reduce the memory footprint of cryptographic operations, including key generation and distribution, encryption, and decryption, to support devices with limited memory resources.
- Streamline the handshake process to minimize latency and improve user experience. Handshake time refers to the time taken to establish a secure connection between communicating devices. Reduce the handshake time compared to traditional TLS setups to minimize latency and enhance the user experience, which is critical for improving responsiveness in real-time communication scenarios.
- Minimize power consumption by optimizing cryptographic operations and reducing computational overhead. Power consumption is a critical consideration, especially in resource-constrained environments such as mobile devices and IoT devices.

1.4 Research Contribution

To the best of our understanding, this research is at the cutting edge, representing a significant advancement in applying the NPF to strengthen security within wM-Bus protocol communications. The distinctiveness and importance of our study are emphasized by several key elements, which we will detail as follows:

1. **Comprehensive Study of Wireless Communication Technologies** offers a significant contribution through a comprehensive study of more than 12 wireless communication technologies, focusing on architecture features such as range, data rate, frequency, protocol structure, and security. This in-depth analysis categorized technologies based on their communication range into short, medium, and long-range connectivity, covering technologies such as Bluetooth, BLE, Z-Wave, ZigBee, WiFi, Wi-SUN, LTE-M, NB-IoT, RPMA, LoRa, Sigfox, and Weightless. The culmination of this research was a high-quality report that provided a detailed examination of each technology, which was further shared with the academic and professional community through a conference paper published at the 2019 IEEE CCECE'19 Canadian Conference on Electrical and Computer titled “A Survey Of Wireless Communications for IoT Echo-Systems” [3] and 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH). IEEE, 2022 ”Demystifying Wireless Technologies for Best Uses in IoT Echo-Systems” [4].
2. **Comprehensive analysis of wM-Bus Standard** The culmination of this research was a high-quality report that provided a detailed examination of each technology,

which was further shared with the academic and professional community through a conference paper published at the IEEE 2022 International Symposium on Networks, Computers and Communications (ISNCC) [5]

3. **Structured Implementation and Evaluation Framework:** details a comprehensive approach to integrating security protocols within a wM-Bus environment. The work involves two parallel streams of implementation:

   - TLS Implementation: A secure layer was developed for wM-Bus using various message types in conjunction with a diverse array of TLS cipher suites. This implementation aimed to assess the applicability and performance of TLS within the wM-Bus context, considering its traditional usage in securing communications.

   - NPF Implementation: The thesis takes an innovative approach by optimizing the NPF for wM-Bus. Twelve patterns within the NPF were implemented, encompassing 16 security properties, including cipher, key exchange, and hash functions. This not only showcases the adaptability of the NPF to the wM-Bus but also its potential for customization to meet specific security requirements.

   The evaluation of both implementations was meticulously carried out across five phases, focusing on three critical metrics: memory usage, packet size, and elapsed time for the handshake. These metrics are pivotal in understanding the impact of the implemented security protocols on the resource-constrained devices characteristic of wM-Bus systems. The structured framework and rigorous evaluation provide a comprehensive insight into the suitability of both TLS and the NPF for securing wM-Bus communications, highlighting the NPF's optimized performance in terms of resource efficiency and operational speed.

4. **Detailed Comparative Analysis with TLS:** In its comprehensive comparison between the NPF and TLS, the thesis provides an in-depth analysis based on the three critical metrics of memory usage, packet size, and handshake elapsed time. This methodical evaluation forms the basis for the argument that the NPF is better suited to the unique constraints of wM-Bus systems than TLS, offering a more efficient use of resources and quicker, more reliable connections.

5. **Optimization of Lightweight Protocols Based on the Efficiency of NX and XX Patterns of the NPF:** This investigation delves into the practical application and comprehensive optimization of the NPF within the wM-Bus environment, with a specific focus on the NX and XX handshake patterns. The research takes a dynamic

approach, moving beyond the execution of predefined patterns. Instead, it integrates extensive security properties directly into the developed Python code, thereby enhancing the system's overall performance, security, and reliability in data transmission. In parallel, the study also encompasses a detailed examination of the TLS protocol by utilizing a combination of eight different cipher suites, three distinct wM-Bus message types, and two different key exchanges. This methodical approach enables a thorough evaluation of the TLS protocol's efficiency and security within the wM-Bus context. By employing identical hardware setups for both the meter and gateway components, the research ensures accurate replication of the wM-Bus functionality, thereby establishing a reliable basis for comparison and optimization. The overarching aim of this phase is to refine and enhance the robustness of the NPF, specifically by tailoring the NX and XX patterns to the unique requirements of wM-Bus communications. This optimization process is aimed at achieving a delicate balance between robust security features and the system's need for lightweight and efficient operation, ultimately contributing to a more secure, reliable, and resource-efficient communication protocol suitable for wM-Bus systems.

Furthermore, In assessing the NX and XX NPF Patterns for their application in wM-Bus environments, we utilized the STRIDE [1] threat model encompassing Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege to thoroughly evaluate their security robustness. This analysis confirmed their efficient balance between security measures and resource demands, establishing their suitability for constrained IoT contexts.

6. **Metric-Based Evaluation for Operational Efficiency:** the thesis provides a detailed analysis of the impact of implementing NX, XX, and TLS protocols on the battery life of IoT devices. This evaluation is crucial in demonstrating the lightweight nature and operational efficiency of the security protocols, especially within the constraints of low-power IoT environments.

   To ensure efficient data transmission as our main objective that the packet size does not exceed 5% comparing to TLS. The packet size for the NPF pattern XX configuration is 209 bytes for the entire handshake process. comparing with TLS, which its about 250 bytes. This means that using the NPF pattern XX results in a packet size that is approximately 16.4% smaller than TLS, under similar conditions. This

---

[1] In 1999, Loren Kohnfelder and Praerit Garg developed threat modeling methodology called STRIDE. This approach aimed to assist Microsoft security professionals in systematically analyzing potential attacks, that could be executed within specific segments of a computer system

reduction is quite significant in bandwidth-limited environments.

Our studies have shown that the implementation of the NX pattern significantly reduces memory usage by approximately 25% compared to traditional TLS approaches. The XX pattern also contributes to this efficiency with a reduction close to 20%, though slightly less optimal than NX due to its more complex handshake dynamics. The comparative analysis highlighted the streamlined cryptographic processes of the NX and XX patterns, which utilize less memory without compromising security.

The NX protocol, with its efficient data handling and handshake processes, was shown to extend the operational longevity of IoT devices to approximately 9.51 years. This surpasses the initial goal of reducing battery consumption compared to TLS, highlighting the protocol's efficiency and lightweight nature. On the other hand, the XX protocol, while still efficient, offers a slightly reduced lifespan of about 7.88 years due to its marginally higher energy demands. Comparatively, the TLS protocol, despite its robust security features, incurs significant energy overhead, limiting the device's operational life to just 3.81 years. These findings underscore the trade-offs between energy efficiency and security functionality in IoT communication protocols. Moreover, Optimizing cryptographic operations to reduce power consumption by at least 15% . Notably, the NX protocol emerges as the most energy-efficient option, providing an operational lifespan close to the industry standard of 10 years for smart meters, with only a minor trade-off in security to achieve a lifespan of approximately 9 years.

1.5 Research Methodology

1.5.1 Introduction

The primary goal of this research is to develop a security and privacy protocol using the NPF for IoT ecosystems, specifically targeting wireless sensor networks (WSNs). Our application focuses on the Advanced Metering Infrastructure (AMI). The research targets critical security and privacy issues in these networks and aims to enhance data transmission efficiency with smaller packet sizes. Additionally, it seeks to minimize the memory requirements for cryptographic operations, reduce power consumption, and lower latency to improve the user experience. by 10% compared to the TLS protocol, making it more efficient and lightweight. The research implementation design will be employed to develop, implement, and test the proposed NPF. The study will compare the performance and efficiency of the NPF within its own patterns to be optimized to be deployed in IoT devices. In addition it will be compared against the existing TLS protocol in terms of security features and battery consumption. The study utilized IoT devices that operate within wM-Bus protocol.

1.5.2 PART I: Wireless Communication Technologies Investigation

1. Compressive study obtained over 12 different wireless communication technologies.
    - A comprehensive study obtained to cover 12 different wireless communication technologies.
      Focused on architecture, features (range, data-rate, frequency, etc.), protocol structure, and security. The research covered LPWAN (Low-Power Long Range) and the study categorize based on the wireless communication range, which it categorized as Short, Medium, and long Range connectivity.
        - Short Range: Bluetooth, BLE, Z-Wave & ZWave Plus, and ZigBee & ZigBee PRO
        - Medium Range: WiFi 802.11a/b/g/n, WiFi HaLow 802.11h, Wi-SUN, and wM-Bus, Dash7
        - Long Range: LTE-M, NB-IoT, RPMA, LoRa, Sigfox, and Weightless
2. Compressive survey conducted for best understanding how communication technologies are best applied to read data from end devices.
   This survey conducted on the leading smart meter vendors. Understanding of how Communication Technologies are best applied to read data from end devices. A survey had been conducted about how Communication Technologies are best applied to read data from diversity products of leading Smart Meter Vendors. In addition to, devices capabilities, Architecture, and Security mechanism.
3. Comprehensive Study of wM-Bus Communication Standards
   The Study commences on an in-depth exploration of the Wireless Meter-Bus (M-Bus) protocol, focusing not only on its structural and technical facets but also on the rationale behind its adoption, particularly in the realm of utility metering and beyond, into industrial sensor applications. This investigation scrutinizes the protocol's packet frame structure, its operational capabilities, and inherent limitations, providing a holistic view of its implementation across diverse environments. Central to this study is the elucidation of the main advantages that make wM-Bus a preferred choice

in industrial settings: its provision of a standardized framework that encompasses a complete communication stack, its optimized design for battery-powered operations, and its efficient two-way communication protocol tailored for sensor data reading and control mechanisms. Moreover, the investigation highlights the widespread availability of compatible data readers and gateways, which underpins the protocol's ease of integration and scalability. By delving into these aspects, the study not only showcases the protocol's pivotal role in promoting smart grid and smart city advancements but also sheds light on its potential to revolutionize sensor-based industrial applications, addressing both its strengths and areas ripe for future enhancements to meet the evolving demands of technological progress and security considerations.

Chapter 3 provides a more detailed explanation of wireless communication technologies investigation

1.5.3 PART II: Research Design - Framework Implementation

The methodology includes a structured execution in three distinct main points, emphasizing the practical implementation and evaluation of the proposed solution. The research introduced as a multi-phase investigation designed to assess and enhance the performance of communication protocols within secure environments.

1. **Design and Architectural of the Proposed Security Solution** This foundational phase involves the conceptualization and architectural design of the proposed security model, leveraging the NPF to tailor a solution that meets the specific requirements of wM-Bus communications. This initial stage involves a detailed design of the proposed model, focusing on creating a clear draw on the communication between two principal entities: the meter (acting as the client) and the gateway (serving as the server). The emphasis will be on:

    - Illustrating the communication flow between the meter and gateway, clearly delineating the roles and responsibilities of each entity within the security framework.
    - Defining the integration points and the role of each protocol within this communication model, particularly focusing on how the NPF enhancesthe security aspects of wM-Bus communications.

    This design phase is critical for laying a solid foundation for the subsequent implementation and evaluation phases, ensuring that the developed protocol is not only theoretically sound but also practically viable.

Chapter 4 provides a more detailed explanation of Lightweight Protocol Design

2. **Framework Implementation and Preliminary Analysis:** The research begins with an essential evaluation of the NPF and wM-Bus protocol, setting foundational benchmarks and providing a thorough understanding of each protocol's standalone performance, with a focus on critical metrics like communication time, packet size, and memory usage. This lays the groundwork for an in-depth analysis of the Noise's functionality in various security setups, examining the effects of encryption methods, key exchange mechanisms, and hash functions through six specific patterns. Concurrently, the study integrates IoT devices in wM-Bus networks to assess the protocol's real-world efficacy, aiming to establish reliable performance benchmarks. This dual-phase approach is vital for identifying the protocols' inherent capabilities and limitations, which is instrumental in devising an effective integration strategy that ensures secure and efficient communication while considering the performance trade-offs inherent in enhancing security.

   Chapter 5 Phase 1 & 2 provides a more detailed explanation of protocols implementation

3. **Simulation, Integration and Optimization** In phases three and four, the research delves into fortifying the wM-Bus protocol with TLS 1.2 to enhance security, assessing the modified protocol's performance against established benchmarks to gauge TLS 1.2's efficacy in improving security without sacrificing efficiency. The study then transitions to a nuanced analysis by integrating the NPF's 12 patterns into the wM-Bus protocol, exploring the impact of both two-message (IK, IN, IX, KK, KN, KX, NK, NN, NX) and three-message (XK, XN, XX) patterns. This thorough examination identifies the most efficient security patterns—NX for two-message configurations and XX for three-message scenarios—based on their impact on security, packet size, and handshake duration, thereby optimizing performance for complex communication sequences.

   Chapter 5 Phase 3 & 4 provides a more detailed explanation of protocols implementation

4. **Evaluation** the evaluation of our proposed solution entails a comparative analysis between 12 optimized NPF patterns to identify the most suitable configurations for integration with the wM-Bus for enhanced security. This involved rigorous testing of the selected patterns against traditional TLS protocols, considering key performance

indicators such as memory usage, packet size, time required for handshakes and data transmission, and battery power consumption. The analysis was comprehensive, extending to both two-message and three-message handshake patterns within the Noise, ultimately selecting the most efficient patterns for detailed comparison. Additionally, the security robustness of the chosen NPF patterns was validated through the application of the STRIDE threat model, ensuring that the integrated solution not only meets but exceeds the security requirements essential for safeguarding wM-Bus communications.

Chapter 6 provides a more detailed explanation of Evaluation

1.5.4 Data Collection and Tools

The research methodology is designed to center on IoT devices deployed within wM-Bus networks, incorporating specialized tools and kits, notably the Radiocraft developer kit (RC1701HP-MBUS4), to intricately simulate the interaction between smart meters and gateway devices. This strategic inclusion of the developer kit is instrumental in facilitating the comprehensive data collection process across all phases of the research. The data collection process is structured to encompass a variety of critical metrics, including memory usage, packet size, handshake execution time, and the security attributes inherent to each protocol, with a particular focus on the NPF patterns. This evaluation is conducted with a dual perspective, taking into account both the end device, represented by the smart meter, and the collector, functioning as the gateway, thereby ensuring a detailed analysis of the system's performance and security efficacy.

In addition to the focused methodology on IoT devices within wM-Bus networks and the utilization of the Radiocraft developer kit (RC1701HP-MBUS4) for simulating smart meter and gateway device interactions, the research also leveraged the comprehensive interface provided by Radiocrafts [6]. This interface was crucial for ensuring seamless communication and configuration of the devices, thereby enhancing the reliability and effectiveness of the data collection process. For the security evaluation of the NPF, the research employed the Noise Explorer online tool [7], a pivotal resource for verifying the robustness and integrity of the protocol's security mechanisms. This tool played a vital role in assessing the cryptographic strength and resilience of the NPF against potential vulnerabilities and threats. Furthermore, the research extensively utilized Python for the entire integration process across all phases. The development was supported by a suite of Python libraries specifically tailored for cryptographic functions, alongside the NPF library. These libraries provided a solid foundation for developing the framework, enabling a streamlined and efficient integration process that was both robust and adaptable to the research's evolving needs. This comprehensive approach, combining specialized hardware interfaces, rigorous security evaluation tools, and versatile software development platforms, ensured a thorough and well-rounded investigation into the application and efficacy of the NPF within IoT ecosystems.

1.5.5 Variables Measures and Evaluation

In the research, we measured variables that are crucial for evaluating the performance and efficiency of the handshake process within the NPF. The elapsed time required to complete the handshake process was a primary variable, calculated based on the outcome of each phase, from the initiation of communication to its completion. This included measuring the time at various stages of the handshake, such as the ClientHello, ServerHello, and ClientFinish.Another critical variable was the packet size, which varied depending on the message type and the specific stage of the NPF handshake process. The variation in packet size is instrumental in understanding the bandwidth efficiency and the data overhead introduced by the protocol at different stages of the handshake.Memory consumption was also a key variable, evaluated to determine the amount of computational resources required to successfully complete the handshake process. This consumption is significantly influenced by the security properties of the protocol, including the choice of cipher, the processes of encryption and decryption, the hash function utilized, the key exchange mechanism, and the length of the keys generated. By analyzing these variables, we were able to gain deep insights into the protocol's resource efficiency and its impact on device performance, particularly in resource-constrained IoT environments. This comprehensive variable measurement approach provided a nuanced understanding of the protocol's operational dynamics and its suitability for secure IoT communications. In assessing the protocol's security, the research conducted a thorough evaluation of its robustness against potential threats using the STRIDE threat model [8]. This model facilitated a structured analysis of various security threats, including spoofing, tampering, repudiation, information disclosure, denial of service. Additionally, the Noise Explorer online tool was employed as a pivotal resource for security verification, providing an automated analysis of the protocol's cryptographic properties. This dual approach, combining a proven security threat model with advanced cryptographic analysis tools, enabled a rigorous and in-depth evaluation of the protocol's security capabilities. Conversely, to assess the protocol's efficiency and lightweight nature, we conducted detailed calculations incorporating a range of assumptions derived from the Radiocrafts datasheet. This approach allowed us to quantitatively evaluate the protocol's resource consumption and operational demands, ensuring its suitability for resource-constrained environments typical in IoT applications. The evaluation of battery life for smart metering within the wM-Bus industry, the standard expectation is a battery life span of 10 to 12 years when operated without any security protocols. However, the integration of the NPF as a security measure has led to a notable decrease in battery life expectancy, bringing it down to 9 years. This is even more pronounced with the implementation of the TLS protocol, where the battery life sharply falls to 3.8 years. To quantify these reductions, the battery life with the NPF sees a decrease ranging from 10% to 25%, depending on the initial expected lifespan (10 to 12 years). In contrast, the TLS implementation results in a dramatic reduction of 61% to 67.7%. Comparatively, the NPF's impact on battery life is significantly less severe than TLS's, aligning with our objective to limit the reduction in battery life compared to TLS. By achieving a reduction far below the 10% threshold when compared to TLS, the goal was successfully met, underscoring the NPF's efficiency in balancing security with battery life.

1.5.6 Consideration and Limitation

This research encountered several limitations that are crucial to acknowledge for a comprehensive understanding of the research's scope and potential areas for future work. A primary limitation was the reliance on two specific devices (cards) to simulate the communication between the meter and the gateway. This simulation, while effective for controlled experimentation, did not encompass the full breadth of real-world scenarios, potentially limiting the applicability of our findings to more diverse and complex environments. In designing the secure protocol within the NPF, we aimed to balance security with practicality and efficiency for use in wM-Bus communication. We ensured the protocol pattern chosen was compatible with the hardware and software platforms of the meter and gateway devices, yet this introduces a limitation in terms of the flexibility and adaptability of our protocol to other systems with different specifications. Furthermore, the protocol was designed with a keen awareness of the resource constraints inherent in meter and gateway devices. This necessitated the use of lightweight cryptographic primitives, optimization for low power consumption, and minimization of message sizes, which, while enhancing efficiency, may restrict the protocol's robustness and adaptability to environments with less stringent resource constraints.

In aligning with the constraints of the wM-Bus network packet frame, we made a deliberate decision to limit the length of messages used within each handshake process. This resulted in our protocol implementation excluding certain cipher suites from the TLS protocol and specific security properties from the NPF that would otherwise require more extensive data exchanges. This decision, while necessary for the focus of this study, presents limitations in terms of the protocol's adaptability to other communication standards or more demanding security requirements.

The refined methodology emphasizes a structured, phased approach to developing and evaluating the proposed NPF. This approach incorporates detailed implementation strategies, evaluation metrics, and cryptographic configurations to enhance security and efficiency in IoT ecosystems. However, it is essential to recognize that this comprehensive methodology is tailored to the specific context of this research and may require adaptation to be applicable in different contexts or to meet other security and efficiency standards.

In summary, designing a secure communication between the meter and gateway utilizing the NPF and the wM-Bus protocol requires careful consideration of the specific requirements and constraints of the communication, as well as the compatibility and interoperability of the two protocols. The limitations and assumptions outlined underscore the importance of further exploration and adaptation to extend the applicability and robustness of the proposed solutions in varied contexts and against evolving security threats.

1.6 Thesis Organization

- Chapter 2: Background and Literature Review - Provides essential background and discusses the security aspects of smart meter infrastructures, WM-Bus in smart metering, and NPF security and implementation.
- Chapter 3: Demystifying Wireless Technologies for Best Uses in IoT Echo-Systems - Explores various wireless technologies, their applications, and specifics of wM-Bus, including a proposed security framework.
- Chapter 4: Lightweight Framework Over wM-Bus for IoT Smart Grid
- Chapter 5: Implementation of Protocols to Secure Wireless Communication - Covers the implementation of the NPF, wM-Bus Protocol, integration with TLS 1.2, and optimizations for a lightweight protocol.
- Chapter 6: Evaluation and Analysis - Focuses on the evaluating of the outcomes of the Implementation, discussing STRIDE threat models, and considerations for battery life.
- Chapter 7: Conclusion and Future Work

Chapter 2

2 Background and Literature Review

2.1 Background

2.1.1 Overview of Smart Grid and IoT Technologies

Smart Grid and Internet of Things (IoT) technologies represent transformative approaches to modernize and enhance the efficiency, reliability, and sustainability of electricity systems worldwide. Smart Grids integrate digital communications and advanced technologies to manage electricity flow more efficiently, accommodating renewable energy sources and enabling real-time monitoring and control. IoT, on the other hand, brings connectivity to everyday objects, turning them into intelligent devices capable of communicating and interacting with each other and with users, thereby offering profound implications for energy management and consumption patterns [9] [10] The integration of IoT in Smart Grids is pivotal for achieving a highly interactive, responsive, and automated energy ecosystem. This integration enables advanced features such as demand response, where devices can adjust their electricity usage in response to signals from the grid, contributing to stability and efficiency. Moreover, IoT technologies facilitate enhanced monitoring and predictive maintenance of grid components, reducing downtime and operational costs [11] [12]. Furthermore, The integration of Internet of Things (IoT) technologies with smart grid systems is revolutionizing the way we monitor, manage, and optimize energy usage. IoT devices, including sensors, routers, gateways, and smart meters, are pivotal in enhancing connectivity and efficiency across the energy sector. These technologies facilitate real-time data tracking, predictive maintenance, and consumption optimization, laying the foundation for a more sustainable and consumer-centric energy ecosystem. The shift towards smart grids represents a critical evolution in energy management, embodying a transition from traditional grids to dynamic, interconnected networks capable of adapting to the changing demands of modern energy consumption [13] [14]. Overall, the synergy between Smart Grid and IoT technologies is essential for creating a more sustainable, efficient, and resilient energy future. As these technologies continue to evolve and mature, they hold the promise of revolutionizing the way we generate, distribute, and consume electricity, paving the way for a smarter, greener, and more interconnected world.

2.1.2 Importance of Communication Technologies

Communication technologies are the backbone of IoT-enabled smart grids, enabling seamless interaction between various devices within the grid. These technologies, which encompass wireless, cellular, and low-cost frequency solutions, are essential for the effective implementation of IoT applications in smart grids. They ensure reliable data transmission, support remote monitoring and control, and facilitate the integration of renewable energy sources into the grid. The role of these communication technologies extends beyond mere connectivity; they are instrumental in achieving energy efficiency, enhancing grid resilience, and empowering consumers with detailed insights into their energy usage [15] [16]. Low-Power Wide-Area Network (LPWAN) technologies, such as LoRaWAN, SigFox WiFi, Bluetooth, NB-IoT, and wM-Bus have emerged as a critical enabler for IoT applications within smart grids. LPWAN technologies are designed to offer long-range communication capabilities with minimal power consumption, making them ideal for IoT devices that need to operate for extended periods on limited power sources. They play a significant role in smart metering, sensor networks, and monitoring systems that contribute to the optimization of grid operations and the integration of distributed energy resources [17] [18]. Moreover, the adoption of LPWAN and other wireless communication technologies in smart grids supports the development of innovative applications such as demand response, asset management, and predictive maintenance. These applications leverage the real-time data provided by IoT devices to make the grid more responsive to changes in demand, enhance operational efficiency, and reduce maintenance costs [19] [20]. Overall, wireless communication technologies, particularly LPWAN, play a pivotal role in the development and success of IoT-enabled smart grids. They provide the necessary infrastructure for reliable, efficient, and secure communication among the myriad of devices that make up the smart grid ecosystem. As these technologies continue to evolve, they will undoubtedly open up new opportunities for enhancing grid performance, reliability, and sustainability.

2.1.3 wM-Bus Protocol

wM-Bus stands out as a critical communication protocol within the Advanced Metering Infrastructure (AMI), primarily used for smart metering applications. Its adoption across Europe for utility metering underscores its importance in the IoT ecosystem, particularly in facilitating efficient utility management. wM-Bus is designed to offer secure, reliable communication between meters and data collectors, enabling the remote reading of utility meters. This protocol is crucial for the operational efficiency of smart grids, ensuring accurate data collection and transmission, which is essential for billing, consumption analysis, and grid management. However, the wM-Bus protocol faces significant security challenges that cannot be overlooked. The primary issue lies not within the protocol or its standard specifications but rather in the implementation of security measures—if they exist at all. Many property management and utility companies opt for either unencrypted meter data transmission or the use of a single encryption key across all meters. This approach, primarily adopted to simplify deployment, reduce computational costs, and minimize security complexity, inadvertently exposes the meters to potential attacks. Research conducted by L. Vegas highlights these security vulnerabilities, including short key sizes limited to 64 bits, zero consumption detection, and plaintext transmission that risks disclosing sensitive information, including encryption keys, thus facilitating man-in-the-middle attacks.

Furthermore, for battery-powered meters, there's a noticeable absence of robust security measures. This gap is particularly concerning given the critical role these meters play in the smart grid infrastructure. While the latest Open Metering System (OMS) specifications have introduced security extensions that offer additional encryption modes and rely on AES-128 with dynamic keys for enhanced integrity, these improvements are not universally applied. Moreover, the reliance on Transport Layer Security (TLS) 1.2 in mode 13, though a step in the right direction, does not fully address the unique challenges posed by the operational and environmental conditions of wireless metering infrastructures.

In Chapter 4 of the thesis, we thoroughly examine all aspects of the wM-Bus protocol, providing a comprehensive overview of its role within the Advanced Metering Infrastructure (AMI) and its adoption for utility metering across Europe. We explore the protocol's network structure, which relies on a star topology with master and slave devices, and its operation across various unlicensed frequency bands. The chapter also covers the protocol stack, detailing the layers from the physical layer up through the application layer, ensuring a thorough understanding of how the wM-Bus functions from the ground up. Moreover, we address the significant security concerns associated with wM-Bus, highlighting the vulnerabilities arising from common practices like using unencrypted meter data or a single encryption key for all meters. Such practices expose the meters to potential attacks, compromising the integrity and confidentiality of the transmitted data. We also introduce the latest security enhancements proposed by the Open Metering System (OMS) group, which aim to mitigate these risks by introducing additional encryption modes and employing AES-128 with dynamic keys.

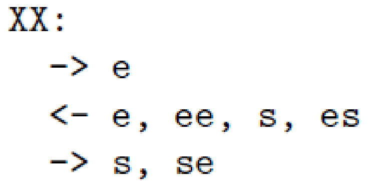
2.1.4 Security Challenges

Security challenges in wireless communication, particularly for IoT devices, and more specifically for those that are battery-powered, represent a critical area of concern in today's interconnected world. For applications like smart metering, where real-time data on energy consumption is vital, these challenges are even more pronounced due to the need for constant, reliable communication and data integrity. IoT devices, by their very nature, are often deployed in vast numbers and in potentially unsecured environments, making them susceptible to a variety of threats. These range from physical tampering and unauthorized access to more sophisticated cyber-attacks aimed at compromising the device's functionality or the data it transmits [21]. The limited computational power and energy resources of many IoT devices further complicate the implementation of robust security measures, making them a weak link in the security chain of the networks they inhabit. In the context of smart meters, which are pivotal for the smart grid's functionality, the security challenges are multifaceted. Smart meters must ensure the confidentiality, integrity, and availability of the data they collect and transmit, all while operating efficiently to maintain long battery life and minimize maintenance. Attacks on smart meters can lead to energy theft, false billing, and even grid instability if the compromised data is used to manipulate energy supply and demand [22] Moreover, the communication protocols used by smart meters, such as wM-Bus, although efficient for low-power wide-area network applications, can be susceptible to eavesdropping and tampering if not adequately secured. Implementing security measures like encryption and anomaly detection can mitigate some of these risks, but they also need to be carefully balanced to avoid significantly increasing the device's energy consumption, which could potentially reduce its operational lifespan. wM-Bus, specifically designed for meter reading systems, requires tailored security solutions that address its unique vulnerabilities while maintaining energy efficiency essential for battery-powered devices. [23] [24] Ultimately, securing wireless communication for IoT devices, particularly in metering applications utilizing wM-Bus, requires a multifaceted approach that addresses the unique challenges posed by the limited resources of these devices and the vulnerabilities of wireless protocols. The development and implementation of lightweight, efficient, and robust security mechanisms are crucial to protecting sensitive data, ensuring the reliability of smart metering systems, and fostering trust in IoT technologies.

2.1.5 NPF

The NPF [2] is emerging as a promising solution to enhance the security of IoT devices and communication within smart grids. This framework employs advanced cryptographic techniques to establish secure communication channels, ensuring data encryption and protecting against various cyber threats. Its application in IoT security and beyond, including in major messaging platforms, highlights its versatility and effectiveness in addressing the complex security challenges inherent in modern digital infrastructures. The NPF stands out for its security and lightweight design, making it highly effective in messaging, Virtual Private Networks (VPNs), and IoT applications by surpassing traditional TLS protocols in terms of efficiency and flexibility, particularly in resource-constrained environments [25]. Its lower computational demand, compared to the more resource-intensive TLS [26], renders it ideal for IoT devices with limited capabilities. The framework ensures secure communication over untrusted networks, effectively safeguarding against eavesdropping, tampering, and impersonation. It is built on fundamental cryptographic components such as symmetric and public-key encryption, digital signatures, and hash functions, and it supports key exchanges, MACs, and padding schemes. The NPF's simple syntax belies the depth of its design, where "tokens" in message patterns enable intricate protocol interactions, as shown in Figure 2.1. The NPF's handshake protocol is crucial for setting up a secure channel between two entities by sequentially exchanging messages. These exchanges not only lay the groundwork for cryptographic negotiations but also authenticate the involved parties and create shared secrets for future encrypted communications. The NPF supports a number of different handshake patterns, each of which is optimized for specific use cases depicted in Figure 2.2. In the context of NPF patterns, "N," "X," "K," and "I" indicate different modes of static key handling during handshakes. "N" represents no static key usage, "K" signifies that a static key is pre-shared, "X" indicates the exchange of static keys, and "I" means immediate static key transmission with minimal identity protection. Patterns like "XX," "IX," "KN," "KX," and "NX" combine these elements to specify particular handshake processes, detailing how keys are exchanged and authenticated. For example, For the "XX" pattern, the cryptographic functions used can vary based on the specific protocol name applied to the base pattern. The specification of the "Noise_NX_25519_ChaChaPoly_Blake2s", which uses 25519 for the Diffie-Hellman (DH) functions, indicating Curve25519 is used. ChachaPoly [27] for the cipher functions, Blake2s [28] for the hash functions.

One example of NPF we present here pattern XX, which is detailed in Figure. 2.1. This particular pattern enables mutual authentication by incorporating both parties'static public keys into the exchange. In this pattern, each party sends its ephemeral public key and receives the other's, performing Diffie-Hellman operations for shared secret derivation. Authentication is achieved as each party sends its static public key encrypted using the derived shared secret, ensuring that only the intended recipient can decrypt and verify the



```
XX:
  -> e
  <- e, ee, s, es
  -> s, se
```

Figure 2.1: NPF's XX Pattern: outlines the message flow for the XX handshake pattern, indicating the sequential exchange of ephemeral keys (e), static keys (s), and the corresponding operations (ee, es) performed between two parties to achieve mutual authentication and establish a secure communication channel.

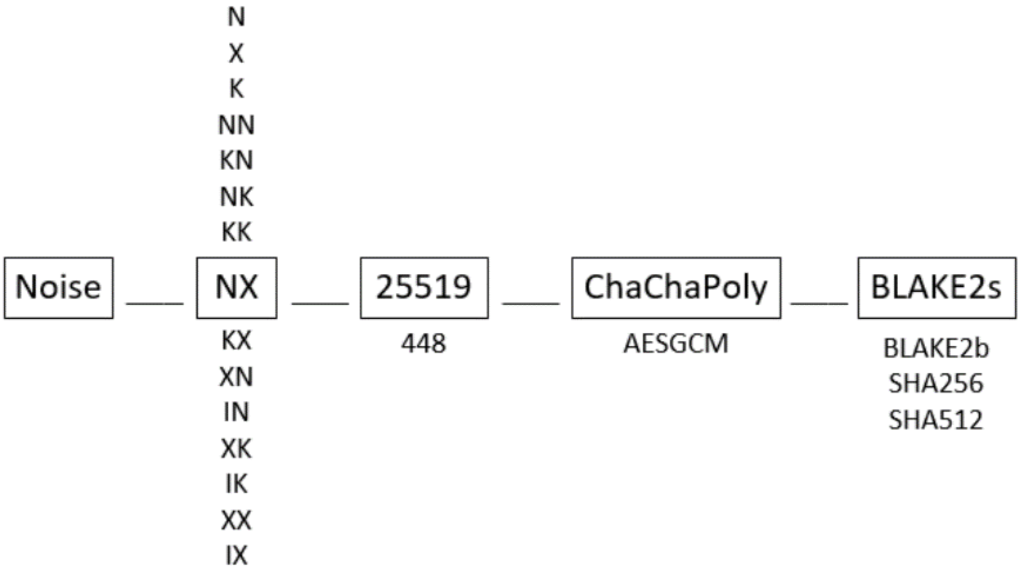


Figure 2.2: Overview of NPF: Handshake patterns and associated cryptographic algorithms

static public keys, thereby confirming the identities of both parties involved in the communication. Initially, the Handshake Pattern XX (Three-Messages) clarifies the pattern token and cryptographic operations of each message, facilitating understanding in later sections. We use 'M' to represent the Meter (Initiator) and 'G' for the Gateway (Responder) in detailing the message flow of the XX pattern:

1. **Meter(M) sends to Gateway(G)**

$$g^m, \quad c_0 = \text{encKDF}_{g^{mg}}(g^m, m_0)$$

2. **Gateway(G) sends to Meter(M)**

$$g^g, \quad c_1 = \text{encKDF}_{(g^{mg}, g^{mG})}(g^m \| g^g, m_1)$$

3. **Meter(M) sends to Gateway(G)**

$$c_2 = \text{encKDF}_{(g^{mg}, g^{mG}, g^{Mg})}(g^g \| c_1, m_2)$$

Where:

g^m	Ephemeral public key of the Meter (Initiator).
g^g	Ephemeral public key of the Gateway (Responder).
g^{mg}	Shared secret generated from the Diffie-Hellman operation between Meter's ephemeral key and Gateway's ephemeral key.
g^{mG}	Shared secret generated from the Diffie-Hellman operation between Meter's ephemeral key and Gateway's static key.
g^{Mg}	Shared secret generated from the Diffie-Hellman operation between Meter's static key and Gateway's ephemeral key.
c_0, c_1, and c_2	Ciphertexts containing the encrypted payloads.
m_0, m_1, and m_2	Plaintext payloads of the messages.
$\|$	Concatenation of byte strings.
encKDF	Encryption function combined with a key derivation function, which encrypts the message and payloads using the keys derived from the Diffie-Hellman operations

Overall, the NPF provides a powerful and flexible tool for building secure communication protocols. Its modular architecture and support for a range of different handshake patterns make it suitable for a wide variety of use cases, from simple peer-to-peer messaging to complex, multi-party applications.

2.1.6 Integration of wM-Bus with NPF

The innovative integration of the Noise Protocol Framework (NPF) with Wireless Meter-Bus (wM-Bus) for securing smart grid communications marks a pioneering advancement in IoT security. This approach is distinctive because, until this research, there has been no precedent for combining the efficiency of wM-Bus communications with the robust security mechanisms provided by the NPF.

The integration addresses several critical security issues in meter-to-gateway communications within smart grids:

- Enhanced Security: The NPF provides advanced cryptographic techniques that protect data integrity, confidentiality, and authenticity. By incorporating these techniques, the integration ensures that the data transmitted over the wM-Bus network is secure from eavesdropping, tampering, and spoofing.
- Efficiency: The wM-Bus protocol is known for its efficiency in data transmission, particularly suited for the resource-constrained environments typical of smart meters. Combining wM-Bus with NPF leverages this efficiency while adding minimal overhead, ensuring that the enhanced security does not compromise the protocol's performance.
- Reliability: The integration offers a reliable communication framework for IoT devices within smart grids. The robust handshake patterns and encryption methods of NPF reduce the likelihood of communication failures and ensure consistent data transmission.
- Energy Optimization: One of the key challenges in smart grid communications is maintaining energy efficiency, especially for battery-operated devices. The NPF is designed to be lightweight and adaptable, optimizing energy consumption while providing high levels of security.
- Scalability: The integration of NPF with wM-Bus supports the scalability required for large-scale IoT deployments in smart grids. It ensures that as the number of connected devices grows, the security and efficiency of the communications are maintained.

This research sets a new standard for securing smart grid communications by effectively combining the strengths of wM-Bus and NPF. It not only enhances the security framework of smart meters but also provides a model for future research and development in integrating lightweight cryptographic protocols with IoT communication technologies.

2.2 Literature Review

2.2.1 Security Aspects of Smart Meter Infrastructures

Smart metering systems, integral to the smart grid, are pivotal in enhancing energy efficiency and management. However, these systems pose significant security challenges due to their critical role in energy distribution and data management. The literature reveals a broad spectrum of research focusing on the security aspects of smart meter infrastructures, emphasizing the need for robust security measures to protect against various threats and vulnerabilities. Orlando et al. [29] present an infrastructure framework for smart grid IoT applications, highlighting the essential role of smart meters in modern energy systems. Their research underlines the importance of integrating secure and reliable communication protocols within smart meter infrastructures to ensure data integrity and privacy. Díaz Redondo et al. [30] focuses on dissecting the security vulnerabilities inherent in smart meter systems. They categorize security threats across physical, network, and application layers, offering a multi-faceted view of potential attack vectors. Importantly, the paper proposes prevention mechanisms tailored to each layer, suggesting a holistic security strategy. For instance, they recommend hardened physical enclosures to deter tampering, encryption for network communications to prevent eavesdropping, and secure coding practices to mitigate application-level exploits. Their layered approach to security provides a blueprint for developing more resilient smart meter systems. Horalek & Sobeslav [31] sets forth a security baseline for substation automation systems, including smart meters. They underscore the importance of a structured approach to security, starting with threat modelling to identify potential risks, followed by vulnerability assessments to pinpoint system weaknesses. The paper emphasizes the implementation of proactive and reactive security measures to defend against identified threats. This includes regular updates to security protocols, rigorous access controls, and real-time monitoring systems to detect and respond to security breaches. Their comprehensive framework aims to fortify the entire substation automation ecosystem against cyber threats. W. Wang *et al.* [32] address smart grid cybersecurity challenges, including cyber threats, interoperability with old systems, standardization gaps, privacy issues, insider risks, resource limits, and advanced threats (APTs). They suggest solutions like robust security, standardized protocols, improved encryption and controls, security audits, and training to counter insider threats, plus stakeholder collaboration. These measures aim to enhance smart grid security and resilience. Nonetheless, the primary hurdles in smart grid security research focus on Generation/Transmission/Distribution, needing efficient attack response, authentication, and key management due to their network size and critical timing for communications. While these issues apply broadly to smart grids, devices like water and gas meters might need specific security approaches due to their unique constraints.

Furthermore, the work by Radoglou-Grammatikis et al. [33] explores the role of firewall systems in protecting smart grid networks, including smart meter infrastructures. The paper highlights the evolution of firewall technologies and their applicability in the dynamic smart grid environment. The authors argue for adaptive firewall solutions capable of responding to the changing threat landscape, offering protection against both external and internal threats. The research points to the need for firewalls that can intelligently manage network traffic, identify malicious activities, and isolate compromised devices to prevent the spread of attacks within the network. Lastly, Kohout et al. [34] address the cybersecurity aspects of smart metering systems, emphasizing the need for rigorous security standards and continuous assessment. Their paper discusses the development of a methodology for testing and validating the security of smart meter systems against a set of predefined requirements. This includes penetration testing, vulnerability scanning, and compliance checks to ensure that smart meters and their associated infrastructure meet the highest security benchmarks. The authors advocate for a cyclical approach to security, where systems are regularly evaluated and updated to counteract emerging threats.

2.2.2 wM-Bus Implementation & Security in Smart Metering

The literature reviews on wM-Bus security in smart metering, particularly for battery-powered smart meters, highlight various challenges and solutions related to the deployment, energy efficiency, and security of smart metering systems.

At first K Zeman *et al.* [35] investigates the integration of wM-Bus technology within the Industrial Internet of Things (IIoT), highlighting its transformative impact on industries through the enhancement of smart device connectivity and communication. The authors illustrate how this technology facilitates efficient and reliable communication critical for Industrial Internet of Things (IIoT) deployments, supported by a practical prototype implementation. In exploring the utility of wM-Bus technology in this context, the paper highlights its adaptability and reliability for industrial communication needs, crucial for the successful deployment of IIoT solutions. Through a prototype implementation, the study showcases the practical benefits of incorporating wM-Bus into IIoT systems, demonstrating its effectiveness in ensuring robust and reliable communication channels in various industrial settings. This research not only provides a thorough technological overview but also bridges the gap between theoretical concepts and real-world applications of IIoT technologies.

By presenting wM-Bus as a competent solution for enhancing connectivity in industrial environments, the paper makes a significant contribution to the field, offering insights into the future of industrial operations in the era of IoT. Furthermore, Pavel's literature *et al.* [36], it focuses on the application of wM-Bus in smart electricity grids, emphasizing its role in enhancing wireless connectivity for smart metering systems. The authors developed a multi-platform framework that serves as a data generator for wM-Bus telegrams, adhering to the EN 13757-4 specification. The real-world evaluation of this framework demonstrates its effectiveness in managing communication distances in indoor scenarios, highlighting its potential in optimizing smart metering infrastructures. Meanwhile, F. Derbel [37] presents a wireless automated meter reading system utilizing battery-powered meters with up to a 12-year battery life, following the wM-Bus standard. It addresses security issues as well as advantages in the metering process and potential added services. furthermore, N. Mochizuki *et al.* [38], explore the use of wireless communication systems in the 920 MHz band for high-capacity IoT/M2M services, including smart metering for gas meters. This study can provide insights into the technological frameworks that could support secure and efficient wireless communication for smart meters.

Although the focus of these studies is more on the energy and capacity aspects of wireless communication for smart meters, they provide a context within which security measures must operate, considering the constraints of battery power and the need for efficient communication.

2.2.3 NPF and its Applications

The NPF is an emerging protocol that is increasingly being utilized to enhance the security of IoT (Internet of Things) devices. As Jalasri *et al.* [39] discussed the concern in collecting data through IoT devices. The concern lies in maintaining data confidentiality, integrity, and privacy, given the risk of unauthorized access during storage. The author presents a solution involving the integration of cryptographic and clustering algorithms within a distributed environment. Data security during transmission is ensured through the NPF encryption method, employing various cryptographic functions for effective data privacy and security. They concluded and suggested future enhancements, such as implementing machine learning models to further optimize clustering and enhance data security. Their proposed system achieved an energy efficiency rating of 99.7%.

NPF is not only gaining traction within the IoT ecosystem but is also finding applications in other domains, including popular messaging platforms like WhatsApp. G. T. Davies *et al.* [40] set up a secure channel between the client and WhatsApp using the NPF. Moreover, Doweling *et al.* [41] focuses on the NPF, which consists of various channel establishment protocols designed to ensure message security while keeping their specifications and configurations simple. Despite its relative newness, NPF is already used in large-scale applications like WhatsApp and Slack. However, the paper highlights that there hasn't been a computational proof of Noise's security properties yet, and it aims to fill this gap. NPF employs a limited set of cryptographic primitives, making it suitable for security proofs based on reduction. The paper mentions that Noise's characteristics as channel establishment protocols align well with the Authenticated and Confidential Channel Establishment (ACCE) model. However, NPF's ability to transmit encrypted messages before full authentication presents a challenge not addressed by the ACCE model. To address this, the paper proposes a generalization of the ACCE model to analyze the security properties of such staged channel establishment protocols more flexibly, similar to the multi-stage key exchange model. In the paper, security proofs are provided for eight of the 15 basic Noise patterns, with a specific focus on proving the security of the XK pattern as an example. lastely, S. Ho *et al.* introduce a verified framework for secure channel protocols, crucial for enhancing communication security across various applications. Their work is pivotal in providing a verified Noise protocol compiler, which can transform any Noise protocol into a secure, high-performance implementation. This advancement underscores the importance of formal verification in cryptographic protocol development, ensuring both theoretical soundness and practical applicability. This research bridges the gap between academic rigor and industry needs by offering a library that guarantees the security of communication protocols, making it a valuable asset for both developers and researchers. The Noise* library stands out by combining the robustness of formal verification with the efficiency needed for real-world applications, marking a significant stride in cryptographic protocol implementation.

After conducting a thorough literature review, it became evident that the NPF could potentially provide an effective solution for securing communication between the meter and the gateway. With confidence in the NPF's capabilities and recognizing the importance of safeguarding this communication domain, we proposed the integration of two protocols: wM-Bus and the NPF. We believe that by combining these two technologies, we have introduced a cutting-edge solution to address the security challenges in this context. Our aim is to ensure that the meter-to-gateway communication remains secure and that this integration represents a state-of-the-art approach in this domain.

Chapter 3

3 Demystifying Wireless Technologies for Best Uses in IoT Echo-Systems

3.1 Overview

The concept of IoT [42] has given researchers a clear objective of ensuring that smart devices are connected to a common platform and able to interact with one another. This objective can be met through establishing a unified communication standard. IoT devices are generally battery powered, and their main requirements include [43] [43]: low energy consumption (long battery life), connectivity (optimized for low data), low cost, low maintenance, and security. IoT devices can be connected through different wireless technologies and use different type of connectivity coverage. Wireless networks such as wireless personal and home area networks (WPANs and WHANs) are short-range networks that use wireless technologies such as Bluetooth, ZigBee, and Z-Wave. Wireless local area networks (WLANs) are medium-range networks with a proximity between 5 and 10 km and are covered by Wi-Fi, WM-Bus, and Wi-SUN. Long-range technologies can expand over 100 km and include unlicensed technologies referred to as low power wide area networks (LPWANs) such as Sigfox and LoRa. There are also licensed technologies such as cellular 2/3/4/5G and LTE. Thus, cellular LTE-M and narrow band Internet of Things (NB-IoT) are known as cellular-IoT (CIoT) technologies. This work investigated recent wireless technologies that support a multitude of coverage ranges in wireless connectivity. This work examined and compared the architecture, security, standard features, and protocol stack of each technology. Subsequently, we highlighted the challenges and security concerns associated with each technology regarding the deployment of IoT domains. We have provided a comprehensive comparison table f or each wireless technology range. In addition, recommendations are provided for the selected technologies illustrated by identifying the most efficient ones that contribute to IoT echo-systems.

The IoT system is valuable due to its exploitation and representation of data. This system must present an architecture that ensures the security, stability, and availability of the IoT. The IoT infrastructure is based on components including devices, gateways, cloud, and applications. Hardware or software devices are often referred to as nodes or end-

---

[0] A version of this chapter has been published in 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE) and 2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH).

devices. The data collector unit (gateway) allows data transfer to the cloud. The cloud creates the mechanism for processing data streams from the gateways or directly from the end-devices. IoT services such as communication, security, device management and data services are essential to support the overall architecture. LPWAN is a type of WLAN designed to connect IoT end-devices that are battery-powered and have low bandwidth. Different wireless technologies with various communication coverage are competing f or a prominent position in the LPWAN domain, as shown in Figure 3.11. However, selecting the most suitable wireless technology is challenging due to the large number of options available, ranging from unlicensed to licensed technologies. The industrial, scientific and medical (ISM) are open frequency bands that differ based on regions and licenses. ISM bands have utilized 900MHz, 2.4 GHz, and5 GHz, but 2.4 GHz is a globally approved band. ISM bands are license-free and consume a minimum of 1 watt of power. However, these bands use spread spectrum technologies, which reduce the power needed to transmit data and increase transmission speed; this allows multiple networks to exist and reduces interference effects. Wireless devices employ different protocol standards to communicate, including proprietary or open standards. The IoT platform was designed to interconnect billions of heterogeneous devices, which requires an adoptable architecture with layered protocol standards. Many wireless protocol standards are developed based on IEEE 802.11 or IEEE 802.15.4g through different structure layers.

IoT security requires securing IoT devices and their networks. However, security aspects are not taken into consideration when IoT devices are manufactured. Institutions must secure end-devices by implementing security standards at different layers and secure protocols for IoT applications via strong user authentication mechanisms and encryption methods. IoT security solutions should be developed within every technology architecture. The main IoT security requirements are data integrity and device security. Digital certification must be enabled to secure devices such as verification and authentication to authorize firmware updates and secure the communication between devices. These security requirements affect multiple protocol layers. Finally, there are trade offs between security and performance, which compromises power consumption [44].

This chapter incorporates research findings that have been published in three academic papers: the 2019 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) paper titled ”A Survey Of Wireless Communications for IoT Echo-Systems” [3], the paper from the 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH) IEEE, 2022 titled ”Demystifying Wireless Technologies for Best Uses in IoT Echo-Systems” [4], and a conference paper presented at the IEEE 2022 Inter-

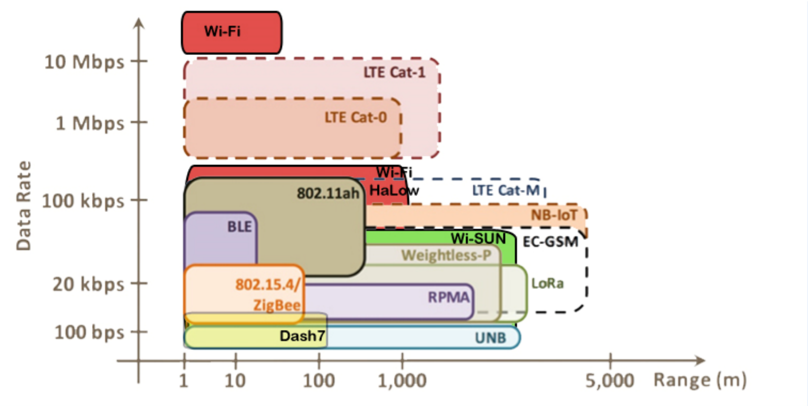


Figure 3.1: Wireless Technology Ranges vs Data Rate

national Symposium on Networks, Computers and Communications (ISNCC) [5].

3.2 Wireless Technologies

3.2.1 Short Range - WPAN and WHAN

Bluetooth and BLE

Bluetooth, based on IEEE 802.15.1, transmits data over 79 channels in the 2.4 GHz band [45], supporting continuous streaming and multi-device communication. Bluetooth ranges include 100 m, 10 m, and 1 m, while BLE extends up to 100 m with a speed of 1 Mbps and larger message capacity (255 bytes)[46]. Security Features include authorization, authentication, and encryption using 128-bit AES for BLE. Key security aspects are pairing and bonding. Bluetooth faces attacks like bluejacking.

Smart Meter Suitability: While Bluetooth and BLE are suitable for short-range communication, they are not ideal for smart meters due to limited range and potential interference in densely populated areas. Additionally, the power consumption of standard Bluetooth is high, and BLE, though more energy-efficient, still falls short in supporting the long battery life required for smart metering applications. The susceptibility to various security attacks further reduces its viability for critical infrastructure like smart meters.

ZigBee and ZigBee PRO

ZigBee is a wireless mesh network standard based on IEEE 802.15.4g, offering reliable, secure, low-power communication with a data rate of 250 kbps. It consists of a coordinator, router, and end devices. The coordinator manages the network and data transmission.

Table 3.1: Pros and Cons of Bluetooth and BLE

Pros	**Cons**
Supports multi-device communication	Limited range for Bluetooth
BLE has low energy consumption	Potential interference
Robust security with 128-bit AES for BLE	Susceptible to attacks like bluejacking
Large message capacity for BLE	Speed and compatibility limitations

ZigBee supports routing and multiple-hop functions for fault tolerance [47]. Security is provided at the MAC, network, and application layers with key management, data protection, and authorization mechanisms [48].

Smart Meter Suitability: ZigBee is suitable for smart meters due to its low power consumption, reliability, and secure communication. However, its limited range and potential for delivery failures due to the tree topology may pose challenges.

Table 3.2: Pros and Cons of ZigBee

Pros	**Cons**
Reliable and secure	Limited range
Low power consumption	Potential delivery failures due to tree topology
Supports routing and multiple-hop functions	Battery optimization relies on beacon intervals
Suitable for smart meter communication	Device placement impacts network throughput

Z-Wave/Z-Wave Plus

Z-Wave is a proprietary protocol developed by Sigma for home automation, operating on low-energy radio frequencies (868.42 MHz in Europe and 908.42 MHz in the US) with a throughput of 40 kbps. It connects up to 232 devices through a central hub using frequency-shift keying and multi-path routing within 100 feet. Z-Wave's four-layer protocol includes application, transport, network, and physical layers [49]. Security features include authentication, confidentiality, and replay attack protection using AES-128 encryption.

Smart Meter Suitability: Z-Wave is not ideal for smart meters due to its limited range and inability to connect to the Internet. However, it offers secure communication suitable for home automation.

Table 3.3: Pros and Cons of Z-Wave

Pros	**Cons**
Low energy consumption	Limited range
Secure communication with AES-128	Not IP compatible
Supports multi-path routing	Works with updated firmware for new security
Enhanced features in Z-Wave Plus	Limited to home automation applications

Table 3.4: Short Range Wireless Technology Specifications

	Standard			**Proprietary**
Specifications	**Bluetooth (4.x)**	**Bluetooth LE**	**ZigBee**	**Z-Wave**
RF/ PHY Layer			US/UE	US/UE
Frequency (MHz)	2.402-2.481 GHz	2.402-2.481 GHz	908/860/2400	908.4/860.42
Number of channels	79	40	2010-01-16	1
Data rate	1 Mbps	200-300 kbps	250 kbps	100 kbps
Channel bandwidth	1 MHz	2 MHz		
Maximum range	10 m	100 m	Indoors: 10-20 m Outdoors: 100 m	Indoors: 30 m, Outdoors: 100 m
Modulation	GFSK	GFSK	BPSK/Q-	FSK, GFSK, narrowband
Addressing / MAC layer	IEEE 802.15.1	IEEE 802.15.1	IEEE 802.15.4	ITU-G.9959
Standardization	Bluetooth Alliance	Bluetooth Alliance	ZigBee Alliance	Zensys, Silicon Lab, Z-Wave Alliance
Nodes per network	1 master + 7 slaves	Not defined; implementation dependent	64,000	Up to 232 devices
Message	31 bytes	255 bytes	133 bytes	64 bytes
Network				
Topology	Scatternet	Star	Star / Mesh	Mesh
Main device type	Master and slave devices	Master and slave device	Coordinator, router, end-devices	Controller and Salve
Nodes to node retransmission	Multi-hop	Multi-hop	32 hops, depending on latency req.	Up to 4 hops
Latency	$\sim$100 ms	$\sim$3 ms	15 ms	$\sim$1000 ms
Routing	Tree based, on-demand route discovery	Tree based, on-demand route discovery	Tree routing, Mesh routing	Source routing
Dedicated Network				
Security	56/128-bit and application layer user defined	AES-128 with Counter Mode CBC-MAC and application layer user defined	128 bit, AES block cipher (CTR, counter mode), authentication CBC-MAC replay protection based on 'counter' and PSK of global default for key exchange	AES-128-CCM authentication algorithm replay protection nonce-based, and using ECDH_curve25519 for key exchange
Applications / logistics				
Power Current	<30mA. sleep: 4μA, Rx: 158.5mA, Tx: 21mA	<15mA	sleep: 3μA, idle: 1.6 mA, Tx: 25.8mA, Rx: 18.5mA	Idle: 1μA , Tx: 20mA, Rx: 40 mA
Battery life	Days	Months to years	100 /1000+ days	1-3 years depends on the device type
Mobility	-	-	-	-
Voice	No	No	No	No
Target application	Mobile phone, computer peripherals, office and industrial automation devices. Wireless headsets, File transfer between devices, Wireless printers, Wireless speakers	Medical devices for monitoring and reporting, Sports and fitness devices, Industrial monitoring sensors, Home automation, Geo-based, targeted promotions via beacons, Public transportation apps, Remote controls, PC peripheral devices like wireless mouse and keyboard	Commercial building automation, Home automation, Industrial plant monitoring, Wireless sensor applications, Smart energy	Residential lighting and automation
Key attributes	Cost, Convenience	Low power	Reliable, Low power, cost effective	
Deployment	Worldwide	Worldwide	Worldwide	Worldwide
Market Maturity	Deployed	Deployed	Deploying	Deploying

3.2.2 Medium Range - WLAN

Wi-Fi

Wi-Fi, released in 1997 by the 802.11 committee, connects devices to the Internet via wireless transmission and radio signals. Operating in the 2.4 GHz and 5 GHz bands, Wi-Fi uses channels to avoid interference [50]. Key standards include 802.11a, 802.11b, 802.11g, and 802.11n. Wi-Fi networks comprise a distribution system, access point, client stations, and wireless medium. Wi-Fi security includes SSID, WEP, and WPA encryption, but is vulnerable to eavesdropping, identity theft, denial of service, and network injection.

Smart Meter Suitability: Wi-Fi is not ideal for smart meters due to high power consumption and vulnerability to wireless attacks, though it provides robust connectivity.

Table 3.5: Pros and Cons of Wi-Fi

Pros	**Cons**
Robust connectivity	High power consumption
Operates on both 2.4 GHz and 5 GHz	Vulnerable to wireless attacks
Supports multiple standards	Potential interference
Wide device compatibility	Not optimized for low-power IoT applications

Wi-Fi Halow

Wi-Fi HaLow, named by the Wi-Fi Alliance, is designed for IoT communications using sub-1GHz frequencies. It operates in the 900 MHz band, which penetrates obstacles better than 2.4 GHz and 5 GHz bands [51]. Wi-Fi HaLow can cover up to 1 km and transmits data at speeds of 150 Kbps on 1 or 2 MHz channels [52]. Despite its capabilities, it is subject to interference from other devices using the 900 MHz frequency[53]. Wi-Fi HaLow ensures strong authentication and encryption but faces new security challenges due to varied implementations by manufacturers. Legacy IoT devices in the sub-1GHz band are not IP-enabled, limiting network integration.

Smart Meter Suitability: Wi-Fi HaLow is suitable for smart meters due to its long-range coverage and efficient penetration through obstacles. However, interference from other devices and the lack of commercially available chips are challenges.

Table 3.6: Pros and Cons of Wi-Fi HaLow

Pros	**Cons**
Long-range coverage (up to 1 km)	Subject to interference
Better penetration through obstacles	Lack of commercially available chips
Designed for IoT communications	New security considerations due to implementation variability
Efficient use of bandwidth with OFDM	Limited legacy product integration

Wi-SUN

Wi-SUN Alliance integrates IEEE 802.15.4g for enhanced utility networks using narrowband wireless technology. It is an open standard for secure, interoperable IoT communications, suitable for smart grid utilities. Wi-SUN supports over 1000 nodes with low-speed communication and operates in star or mesh topologies [54]. It provides a maximum data rate of 300 kbps and a minimum latency of 0.02s. Security includes Extensible Authentication Protocol, 802.1x standards, and AES-CCM encryption.

Smart Meter Suitability: Wi-SUN is highly suitable for smart meters due to its secure, low-power, and scalable nature, making it ideal for large-scale deployments.

Table 3.7: Pros and Cons of Wi-SUN

Pros	**Cons**
Supports over 1000 nodes	Limited data rate (300 kbps)
Low power consumption	Complexity in managing large networks
Secure communication with AES-CCM	Requires advanced infrastructure
Suitable for large-scale smart grid applications	Implementation and maintenance costs

WM-Bus

The Wireless Meter Bus (WM-Bus), defined by the Open Metering Systems group (OMS), is primarily used for remote reading of gas, water, heat, or electricity meters and is also suitable for industrial wireless sensor networks (IWSN). WM-Bus operates in the 858 MHz, 169 MHz, and 433 MHz ranges, supporting long-range communication with low energy consumption. It uses a star topology network with master and slave devices, offering data rates from 2.4 kbps to 100 kbps and ranges up to 2,000 m at 169 MHz. WM-Bus has limited security features [45] which support only AES-128 encryption but no other mechanisms such as device authentication and key distribution [55]

Smart Meter Suitability: WM-Bus is highly suitable for smart meters due to its low power consumption, long battery life (up to 10 years), and robust communication range. However, it has limited security features.

Table 3.8: Pros and Cons of WM-Bus

Pros	**Cons**
Low power consumption	Limited security features
Long battery life (up to 10 years)	Key size limited to 64 bits
Long communication range (up to 2,000 m)	Vulnerable to ciphertext and IV manipulation
Supports unidirectional and bidirectional communication	Predictable IVs
Suitable for AMI metering infrastructure	Privacy issues due to information disclosure

DASH7

DASH7 (D7), promoted by the DASH7 Alliance [56], is the ISO 18000-7 standard for active RFID and wireless sensor networking. It operates in the sub-GHz ISM/SRD band at 433.92 MHz, suitable for low-power, low-bandwidth digital communications with batteries lasting for years. D7 offers adjustable ranges from 10 meters to 10 km and uses less than 1mW of power, penetrating walls and concrete. It supports multi-hop, mesh, and peer-to-peer (P2P) networking, and integrates with other technologies like LoRaWAN, BLE, and NB-IoT. D7 supports MAC data integrity and AES128 EAX cryptography but has vulnerabilities in CRC validation. The file system configures roles like root, admin, and guest. D7 faces issues with limited implementations, support for only 2 hops, no TCP support, and inefficient push communication, as sensors continuously send data.

Smart Meter Suitability: DASH7 is suitable for smart meters due to its long range, low power consumption, and ability to penetrate obstacles. However, it has limitations in high bandwidth data transfers and security concerns with its CRC validation.

Table 3.9: Pros and Cons of DASH7 (D7)

Pros	**Cons**
Long range (10 meters to 10 km)	Limited throughput (200 kbps)
Low power consumption	Vulnerable CRC validation
Penetrates walls and concrete	Few implementations available
Supports multi-hop, mesh, and P2P networking	No formal TCP support
Integrates with LoRaWAN, BLE, and NB-IoT	Sensors always sending data

Table 3.10: Medium Range Wireless Technology Specifications

Specifications	**Wi-Fi**	**Wi-Fi HaLow**	**Wi-SUN**	**WM-BUS**	**Dash7**
RF/PHY Layer					
Frequency (MHz)	2.4 GHz, 5GHz	900 MHz	Unlicensed Sub-GHz RF	Unlicensed 868 MHz, 433 MHz, 169 MHz	433.92 MHz
Number of channels	64	-	-	-	1-8
Data rate	54 Mbps	150 kbps	100/150 kbps	2.4/4.8/19.2 kbps	28kbps/200kbps
Channel bandwidth	14 (2.4 GHz)	1, 2, 4, 8, and 16 MHz	400 kHz (use 2 channels)	19.2 kbps	27.8 kbps
Maximum range	50-100 m	Up to 1 km	1-5 km	500m (868MHz) and 2000m (169 MHz)	250 m
Modulation	BPSK, QPSK, COFDM, CCK, M-QAM and DSSS	OFDM with BPSK, QPSK, QAM, 16-QAM, 64-QAM or 256-QAM	FSK	FSK/GFSK/MSK/OOK/ASK	FSK/GFSK
Duplex	Half-duplex	Half-duplex	bi-directional	Half-duplex	-
Addressing / MAC layer	IEEE 802.11a/b/g/n	IEEE 802.11ah	IEEE 802.15.4g		ISO/IEC 18000-7
Standardization	IEEE 802.11 working group	IEEE 802.11 working group	Wi-SUN Alliance	M-Bus Alliance	
Nodes per network	unlimited	8,191	>1 million	Not specified	
Message	0-2312 bytes	250 bytes	0-64 bits	Not specified	-
Network					
Topology	Ad-hoc, star, mesh	Star, Tree	Mesh, Star	Star	-
Main device type	Node, Base station	Node, Base station	Node, router node, border router	Master, Slave	
Nodes to node retransmission	-	Tree-based multi-hop network	-	-	Multi-hop
Latency			0.02s	-	2 s
Routing			Discovery and Join, Optional Mesh Under routing	-	-
Dedicated Network			No	No	
Security	AES 32 bit and CR4	WPA	802.1X/EAP-TLS/PKI Authentication. Frame security is implemented as AES-CCM* as specified in IEEE 802.15.4	AES128 Counter Mode (AES-CTR)	AES128, Public Key
				, AES128 Cipher Block Chaining Mode	
				(AES-CBC) with dynamic IV	
Applications / logistics					
Power Current		10μW	Sleep: 2μA, Listening: 8mA	Sleep: 0.6μA, Tx: 45mA, Rx: 31mA	Sleep: 4μ Tx: 31mW Rx: 7.5mA
Battery life		10 years			10 years
Mobility					
Voice	Yes	No	No	No	No
Target application	Wireless LAN connectivity, broadband internet access	Sensor networking, industrial automation, home automation, utility networking, extended range Wi-Fi	Advanced metering infrastructure, distribution automation, intelligent transport and traffic systems, street lighting, and smart home automation.	Suitable for remote meter reading, meter maintenance and configuration	
Key attributes	High data rate, short range	Great distance coverage, high device density per AP, high uplink and downlink throughput, and requires no network usage fees to be paid to wireless telecom companies.	Lower power consumption, low data rate, high node density.	Large number of connectable devices, possibility for network expansion, failsafe characteristics/robustness, minimum power consumption, and acceptable transmission speed.	
Deployment		Not standardized, no products yet available			
Market Maturity	Deployed	Deploying	Deploying	Deployed	Deployed

3.2.3 Long Range - WWAN

LoRa

LoRa, designed by Semtech, is a proprietary wireless technology using SSD modulation in the Sub-GHz band, offering long-range coverage (up to 10 miles) and long battery life. It supports 1 million nodes and has localization capabilities, making it ideal for connecting sensors to the cloud for real-time communication. Operating frequencies include 902-928 MHz and 863-870 MHz. The network packet size is 256 bytes with a data rate of up to 300 kbps. LoRa uses chirp spread spectrum modulation at the PHY layer and supports bidirectional communication in a star topology.LoRaWAN, an open-standard protocol for LPWAN, provides encryption and signing for security. Devices activate via personalization or over-the-air methods. Issues include limited dynamic range, no low-power network sync standard, proprietary PHY layers, Semtech-only transceivers, and high downstream latency.

Smart Meter Suitability: LoRa is suitable for smart meters due to its long-range coverage, low power consumption, and robust security features. However, it has limitations in dynamic range, network synchronization, and high downstream latency.

Table 3.11: Pros and Cons of LoRa

Pros	**Cons**
Long-range coverage (up to 10 miles)	Limited dynamic range
Long battery life	No standard for low-power network sync
Supports 1 million nodes	Proprietary PHY layers
Robust security with encryption and signing	Transceiver only available from Semtech
Suitable for real-time communication	High downstream latency

Sigfox

igfox is a global network operator for low-power devices that transmit small data packets (12 bytes). It offers long-range coverage, long battery life, and low-cost end-devices, operating in unlicensed ISM bands with a proprietary protocol. Sigfox transmits up to 140 messages daily per device at a data rate of 100 bps, suitable for applications with low and irregular transmission. It uses ultra-narrow band or narrow band at the PHY layer. Sigfox lacks built-in message encryption and key exchange, requiring vendors to create their own encryption schemes. It uses unique keys for message identification and end-to-end authentication between devices and the cloud. Security relies on stored keys and unique message signatures. Sigfox does not support two-way communication or transmission acknowledgment.

Smart Meter Suitability: Sigfox is suitable for smart meters due to its long battery life and low cost. However, it lacks message encryption and key exchange, and it does not support two-way communication or transmission acknowledgment.

Table 3.12: Pros and Cons of Sigfox

Pros	**Cons**
Long-range coverage	No message encryption
Long battery life	No key exchange
Low cost end-devices	No two-way communication
Suitable for low and irregular transmission	No transmission acknowledgment
Operates in unlicensed ISM bands	Subscription fees required for deployment

LTE-M

Long-term evolution (LTE-M) is designed by 3GPP in the Release 13 specification as a low-power, wide-area technology. It supports IoT devices with long-range coverage and a battery life of up to 10 years, using existing LTE infrastructure. LTE-M operates at 1.4 MHz with uplink and downlink speeds up to 1 Mbps and a latency of 10-15 ms. It includes power-saving modes and offers user authentication, data confidentiality, integrity, and device identification.

Smart Meter Suitability: LTE-M is suitable for smart meters due to its long battery life, robust security, and efficient operation. However, high power usage during firmware upgrades and OTA updates can be a consideration.

Table 3.13: Pros and Cons of LTE-M

Pros	**Cons**
Long battery life (up to 10 years)	High power usage during firmware upgrades
Utilizes existing LTE infrastructure	High power usage during OTA updates
Robust security features	
Supports mobile and higher throughput devices	
Efficient operation with power-saving modes	

NB-IoT

Narrowband IoT (NB-IoT) is a cellular communication solution within the LPWAN domain, developed using LTE standards with DSSS modulation. It offers good coverage, including deep indoor penetration, and operates in the 700 MHz, 800 MHz, and 900 MHz bands. NB-IoT is easy to deploy in existing cellular networks, supports more than 100,000 devices per station, and has an extended range in buildings and underground. It uses 180 kHz bandwidth, with data rates of 200 kHz narrow band.

Smart Meter Suitability: NB-IoT is suitable for smart meters due to its good coverage, deep penetration, and support for many devices. However, it has some limitations in security and latency.

Table 3.14: Pros and Cons of NB-IoT

Pros	Cons
Good coverage and deep penetration	Partial acknowledgments
Supports more than 100,000 devices/station	Long latency
Easy to deploy in existing networks	Data discovered once outside NB-IoT network
Extended range in buildings and underground	can't have a dedicated network infrastructure
Utilizes LTE-based authentication and encryption	

RPMA

Random Phase Multiple Access (RPMA) is a wireless technology designed for IoT and M2M applications by Ingenu. It operates in the globally available 2.4 GHz ISM band using Direct Sequence Spread Spectrum (DSSS) for uplink communication. RPMA offers low power consumption, excellent in-building range, and supports devices operating for over 10 years on a single charge. It provides download speeds of 31 kbps and upload speeds of 15.6 kbps, making it ideal for remote or hard-to-reach locations. RPMA uses AES 128-bit encryption and supports mobility.

Smart Meter Suitability: RPMA is suitable for smart meters due to its long battery life, excellent in-building range, and robust security features. However, higher frequencies may result in increased propagation loss and structural penetration issues.

Table 3.15: Pros and Cons of RPMA

Pros	Cons
Low power consumption	Increased propagation loss at higher frequencies
Excellent in-building range	Structural penetration issues at higher frequency
Long battery life (over 10 years)	Higher processing power usage
Robust security with AES 128-bit encryption	Uses more processing power which may result in lower battery life for some applications
Supports mobility	

Table 3.16: Long Range Wireless Technology Specifications

	Standard	Proprietary	Standard	Standard	Cellular IoT	Cellular IoT	Proprietary
Specifications	**LoRa**	**Sigfox**	**Weightless**	**Weightless**	**NB-IoT**	**LTE-M**	**RPMA**
RF/ PHY Layer		EU/US	-N	-P	EU	US	USA
Frequency (MHz)	863 MHz, 902 MHz, 779 MHz	868 Hz/902 Hz	200 Hz	868 MHz, / 12.5 kHz	200 kHz	LTE bands 450-2350	2.4 GHz
Number of channels	10 EU, 64 US	360	8	8			40
Data rate	0.3-37.5 kbps	0.1 kbps 0.6 kbps	30-100 kbps	Up to 100 kbps (adaptive)	360 kbps	1 Mbps	624 kbps
Channel bandwidth	125 kHz - 250 kHz	1.5 kHz	200 Hz	12.5 kHz	180 kHz	1.4 MHz	1 MHz
Maximum range	15 km rural, 5 km urban	30 km rural, 10 km urban	5 km	2 km	15 km	11 km	1-3 km (urban)
							25-50 km (rural)
Modulation	CSS, DSS with chip	UNB / DL:GFSK / UL:DBPSK	UNB DBPSK	GMSK, offset-QPSK	OFDMA	OFDMA	DSSS
Duplex	HDX (uplink)	HDX (limited) Mainly uplink	Time-division duplex	Time-division duplex	half-duplex	half-duplex	Half-duplex (uplink)
Addressing / MAC layer							IEEE 802.15.4k
Standardization	Unlicensed - LoRa Alliance	Unlicensed - ETSI released (Sigfox company)	Licensed-Weightless SIG	Licensed-Weightless SIG	Licensed - 3GPP Rel.13	Licensed - 3GPP Rel. 13	Ingenu (formerly OnRamp)
Nodes per network	40,000	50,000	Unlimited	Unlimited	200,000	20,000 / AP	500,000 (384,000 per AP)
Message	19-250 bytes	12 bytes, 140 msg/day	20 bytes	10 byte	UL:125 bytes, DL:85 bytes		6-10 bytes
Network							
Topology	Star, Mesh	Star	Star	Star	Star	Star	Star, Tree (PRMA extender)
Main device type	Node, gateway, network server, network application	Sigfox base station, Sigfox cloud, Sigfox node, client server	Terminal, cloud, database, client server	Terminal, cloud, database, client server			
Nodes to node retransmission	6-12 hops	Multi-hop	Multi-hops				
Latency	1-10s	1-30s	Not specified but very high	Not specified but very high	1.5s - 10s	100ms	>20s Depends on architecture
Routing	Route discovery, selection, maintenance, data forwarding and representation				Multicast, firmware broadcast.		
Dedicated network	Yes	Yes	No	No	No	No	
Security	AES 16bit, HMAC	Encryption not supported	AES 128b			3GPP (128-256bit)	16b hash /AES 256b
Applications / logistics							
Power Current	Sleep:2μA, Tx:28mA, Rx:10.5mA, listening:12mA	Sleep:<4μA Tx:45mA, Rx:10mA	Sleep:<4μA, Tx:49mA, Tx:13mA		Sleep:1μA Tx:45mA, Rx:10mA	Sleep:8μA, idle:9mA, Tx:100-490 mA, Rx:n/s	
Battery life	10 years	7 years	10 years	3-8 years	6 years	5 years	10 years+
Mobility/Locality	Yes	Limited mobility/ no localization	yes		No	limited	Yes
voice	No	No	No		Limited	No	No
Target application	Industrial automation, monitoring, sensors and smart metering	environmental sensors, smart meters, patient monitors, security devices and street lights	Automotive, healthcare, media tablets, asset tracking, smart grid, POS		Idle for smart parking, smart meters, smart trash management, and smart sensor applications	Automotive & transportation, Energy industrial, residential, healthcare, Alarm monitoring, asset monitoring.	Digital oilfield, connected cities, usage-based insurance, agriculture.
Key attributes	Private/public network, good hardware availability, MAC and Network layers open.	Long rage, low power, low cost, cloud architecture	Adaptive data rate, high quality of service.		Extreme low power, deep in building and underground propagation	Low power, low cost, LTE stability and latency, added features (voice, roaming, power), and tunable parameter for (coverage, power management, scale)	High uplink rate, offer extreme coverage, high capacity.
Deployment	Early trials & deployments by some operators. Several. operators are members of	Network deployed & running in several countries. Several	No deployments so far	No deployments so far	Initial regions Europe	Initial regions North America Promising as this being a	Deployments in several countries. Not much Insight into transition plan should MNOs find it infeasible to run the network. Transitioning will entail replacement of end-point communications module in the endpoints
Market Maturity	Deploying	Deploying	Deploying	Deployed	2018 Launch	2017 Launch	Deploying

3.3 Wireless Technologies For Echo-System Applications

3.3.1 IoT Echo-System Requirements

IoT comprises a wide variety of wireless technologies across an interminable array of applications. Billions of devices and countless products and services can be connected to the Internet, however there is little mention of the underlying technologies involved and why one is better than another for any specific IoT application. An explanation as to why so many IoT options exist is that there are many industry applications, as the range of application requirements vary from one domain to another. In order to choose the right technology, certain considerations must be taken into account; Power consumption, range and proximity, number of devices, message size, time-sensitivity, security, and of course cost. As has been demonstrated in this survey, a number of wireless technologies are available. Anticipated requirement "Size, Weight, Power and Cost" or SWaP-C, could be the answer for the right wireless technology for the desired IoT application. As shown in 3.17, the suitable technology for each echo-system application is based on their technical specification and IoT requirements. For example, if a higher data rate is needed for the application, NB-IoT can be used. If cost is a priority and the application does not require a high data rate, then LoRa is a good fit, as it offers a private network without the need to depend on a provider. Sigfox and LoRaWAN will serve the lower device cost, provide a very long range (high coverage), infrequent communication rate, and very long battery life. One of the differences between Sigfox and LoRa is that Sigfox is based on UNB sub-GHz while LoRa is based on wide band Sub-GHz. The wide band solution is prone to interference from UNB technologies. Sigfox suffers from a single point failure due to the fact that all infrastructure is owned by Sigfox. One-way of communication (uplink), and low data rate limits the scope of possible applications. Conversely, Wi-SUN is very low in power usage with a low data rate, yet the cost of the devices could pose an issue. Some technologies require a monthly fee based on data usage and number of devices. On the other hand for the short range technologies such as Bluetooth LE is aimed at a completely different set of applications. The technology supports lightning-quick connection, transmission of short bursts of tiny packets of data, followed by lightning-quick disconnection. Many IoT applications have become very popular because of ZigBee, and WiFi. The idea of a mesh network has been very popular over the last five to seven years though many companies adopted it because there was no other choice. Many low power radio technologies coming out today are designed specifically for IoT, and for many uses, a mesh topology is not the best choice. As for Z-Wave, it is similar to ZigBee, the main difference between the two is the data throughput, Z-Wave is roughly 6 times slower than ZigBee. It does, however, require less energy to cover the same range as ZigBee. Cellular communication could be a good potential for smart metering, the main advantage of the cellular technology being that the infrastructure is already there, it can be very power-efficient, and has fast data rate. Yet, the concern is the availability of the technology. Some are promising, yet there are no real deployments. NB-IoT, another CIoT technology, is great indoors, has a low power usage, and is expanding worldwide. It also has a great battery life, deep penetration in building and underground propagation. Moreover, devices are medium-cost and cannot have a dedicated network infrastructure. Security is also an essential element that must be considered, as wireless technologies can be received by anyone. Security requirements depend on the answers to the following questions; How sensitive is the transmitted data? How secure must it be? What authorizations and authentication mechanisms are in place? And finally, what encryption mechanisms are in place during data transmission that guarantees the data's integrity? Wireless technologies are vulnerable to different types of attacks such as eavesdropping, Denial of Service (DoS), Man in the Middle (MIM), spoofing, Replay, etc. There is a certification for each LPWAN technology to ensure their devices/solutions are compatible with other devices within the Echo system.

3.3.2 Wireless Technologies for Smart Metering

Technology is paving the way for the formation of clean energy initiatives including the smart grid. As this efficient, intelligent delivery of energy evolves, users and utilities will enter into a two-way communication model that will allow smart meters to provide real-time energy consumption data directly to the user for up-to-date monitoring. The next step in this clean energy initiative is to identify the importance and increase the use of wireless technology, which would allow users to remotely monitor and control energy use. In order to achieve the next step, many technologies address smart energy management. Some are still under development other at the deployment process. What is helping Advanced Metering Infrastructure (AMI) is power efficiency, bandwidth and latency benefits. Smart metering aims to reduce energy consumption and costs. Governments, regional regulatory bodies, energy/utilities sectors, system integrators,design houses and original equipment manufacturers (OEMs) are involved in worldwide deployments of telemetry infrastructure. This is used by utilities in residential, commercial, and industrial scenarios. There are many different approaches for linking intelligent energy meters to create 'smart' grids, and wireless technology is a key design option. Using a wireless link can simplify the installation and collation of data from a smart meter back to a hub either in the home or in the

Table 3.17: Wireless Technology Suitable Uses for IoT application

Technology	**Suitable use cases**
Bluetooth	Designed to operate in an environment of many users. Up to eight devices can communicate in a small network called a piconet. A good application for Bluetooth LE is one in which the Bluetooth radio will be switched off almost all the time.
BLE	Supports lightning-quick connection, transmission of short bursts of tiny packets of data, followed by lightning-quick disconnection. Suitable for blood pressure monitors, industrial monitoring sensors, and public transportation apps.
ZigBee	Intended for embedded applications with low power consumption and low data rates. Mobility is a constraint; it is not suitable with radio equipment. Typical application like automation systems, wireless sensors networks, industrial control, smoke alarms, and medical data collection
Z-Wave	Simple installation, low power consumption, remote or local control
Wi-Fi	Number of devices connected to a Wi-Fi access point or the distance of device to access point is limited.
Wi-Fi HaLow	Access point could associate more than 8,000 devices within a range of 1 km, making it ideal for areas with a high concentration of things. Suitable for metering, environmental monitoring, industrial sensors, home & building automation, and healthcare.
Wi-Sun	Lower power consumption, low data rate, high node density. Suitable for advanced metering infrastructure, distribution automation, intelligent transport and traffic systems, street lighting, and smart home automation.
WM-Bus	Large number of connectable devices, possibility for network expansion, fail-safe characteristics/robustness, minimum power consumption, and acceptable transmission speed. Suitable for remote meter reading, meter maintenance and configuration.
DASH7	Fills the gap between the short and medium area networks. D7 excels in urban and industrial network installations connecting actuators and messaging applications (sensors, alarms, states) with ranges up to 500 m
LoRa	Regional coverage with low duty cycle and low data rate for long battery life. Different requirements per country. Suitable for private network without relying on a provider. Support mobility applications.
Sigfox	Regional coverage with very low duty cycle and very low data rate for very long battery
RPMA	High uplink rate, virtually unlimited scalability, globally available interference-proof license-free band.
Weightless-P	Adaptive data rates, open standard, high quality of service. Extremely low energy consumption in the idle state helps achieve 10-year battery life.
LTE-M	For places with unlimited network connectivity. Superior data rates, decent range, low power use, best security
NB-IoT	Worldwide connectivity with very good coverage and medium data rate. Consumes more power due to the long transmission.

street, but such links need to be cost-effective and also secure. This has a significant impact on a choice of the many different frequencies, protocols and topologies available for providing the smart meter link. Utilities need robust and flexible communication systems to meet diverse network requirements throughout their service territory. It is a given that communication is central to the basic concept of the smart grid, but there is no “one-size-fits-all” communication technology, options include private RF mesh solutions that connect meters via a concentrator; point-to-point (under glass) communications with individual meters using public cellular networks (which also provide the backhaul for mesh networks); Power Line Communications (PLC); Wi-Fi; and several others. In fact, the majority of utilities in North America today use public wireless networks (i.e., networks owned by carriers such as AT&T or Verizon Wireless) to backhaul metering information collected at the neighborhood-area network (NAN) level to a central location. Today, perceptions about using cellular technology all the way to the meter are beginning to change, as public carriers work closely with operators and vendors to introduce more attractive rate plans and demonstrate that they can meet the utilities’ stringent requirements. Using spread spectrum (DSSS) technology avoids problems in noisy electromagnetic environments. Mesh networks can provide significant advantages for design and implementation of wireless nodes for smart meters. Reducing power consumption to extend the battery replacement cycle as long as possible, or even eliminating it entirely by using energy harvesting, can provide cost benefits as operators roll out smart grid systems. The choice of frequency depends on the range requirements, but many transceivers provide significant flexibility to support a wide range of bands within a single design without needing extra external components. Another important aspect of smart meter design is security. Security is a fast-evolving landscape and the complex IT networks that utilities companies deploy will need to operate for a very long time. Security will therefore require continued attention throughout a network’s lifetime.

3.4 Wireless Meter Bus: Secure Remote Metering within the IoT Smart Grid

3.4.1 Overview

The ”smart grid” is currently considered an important stage for energy systems, which have been enhanced by utilizing communication technologies and various ranges of connectivity to employ smart resources. Smart grid technologies include lots of IoT devices such as sensors, routers, gateways, and smart meters. All these devices facilitate connectivity and communication and enable consumers to optimize their energy use. A smart grid is often defined as a self-sufficient distributed system.

Industrial organizations have recently been exploring the many opportunities offered by deploying the Internet of Things (IoT) in the utility domain. IoT has immense potential to enhance efficiency, improve energy usage, and provide better customer service. In addition, communication technology has made wireless, cellular, and wireless frequencies low-cost and easy to implement in any smart grid application. Smart meters, such as water, gas, and electric meters, are among the main components used in smart grids. Smart meters provide a clear understanding of energy consumption habits.

In general, smart meters allow end-users and utility companies to collaborate and be able to monitor their daily power consumption, thereby reducing their bills. Smart meters enable two-way communication between the meter and the data unit collector (DUC)/gateway. Numerous technologies are utilized when implementing a network for a metering system, such as short- to long-range wireless communication. An example is a low-power wide-area network (LPWAN). Specifically, LPWANs for water and gas meters are battery-powered which makes power consumption crucial. To achieve the best compromise between power utilization and communication range, meters were designed to choose radio frequencies in the sub-GHz bands, as North America uses 915- MHz bands, while Europe uses 868-MHz, 433-MHz, and 169-MHz bands. Yet, most smart meter manufacturers prefer the 2.4-GHz worldwide frequency band. Smart meters, such as gas and water meters, are battery-powered within the wireless network, making meter monitoring a difficult task since it requires working for 10–15 years with a limited energy source. In recent years, wM-Bus, a wireless communication technology, has emerged as one of the most beneficial wireless protocols with AMR.

In this chapter, we highlight about the contribution's state of the art proposes implementing a new security profile called 'W'. Currently, OMS standards use only three profiles: A mode 5, B mode 7 and C mode 13 (specific for Germany). Our proposed implementation of the new profile covers mode 9 AES-128- GCM (authentication is GMAC, MAC is 12 bytes). Our proposed framework offers the main security factors, integrity and confidentiality/privacy. Moreover, the framework considers offering a lightweight protocol to save on the power consumption of the meter battery life.

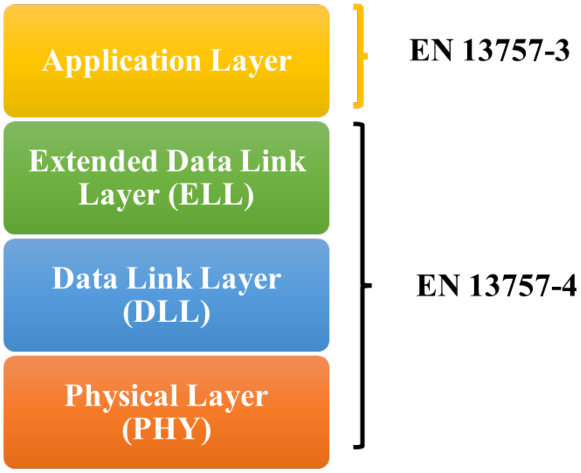


Figure 3.2: Wireless Meter Bus (wM-Bus) Protocol Stack

3.4.2 wM-Bus

wM-Bus Overview

The Wireless Meter Bus (wM-Bus) is an open standard created for smart metering and AMI applications [3] and formed by the Open Metering Systems (OMS) group [57]. Its use is rapidly spreading in Europe for electricity, gas, water, and heat metering. A wM-Bus network is based on a star topology network with master and slave devices described in the EN 13757 standard. wM-Bus can be categorized as short or medium within the wireless communication technology range. The wM-Bus protocol is standardized (EN 13757- 4) and operates on unlicensed 169, 433, and 868-MHz frequency bands [35].

wM-Bus Protocol Stack

The wM-Bus protocol stack is comprised of the physical layer (PHY), data link layer (DLL), extended data link layer (ELL), transport layer, and application layer (APL), as depicted in figure 3.2.

- **EN 13757-3:** In the application layer, the third part describes a standardized application protocol to enable the multi-vendor cooperability. Thus, devices from various manufacturers can be combined into a single system.
- **EN 13757-4:** Wireless meter readout operates the 868-MHz to 870-MHz SRD band. This section identifies the wireless communication of M-Bus and covers the implementation for each layer in the protocol stack. The standard includes a physical layer as well as a data link layer for using wireless devices, and it corresponds to specifi-

Table 3.18: Wireless M-BUS Transfer Modes

	Mode		**Comm.**	**Freq MHz**	**Speed kbps**
S	Stationary	S1	Unidirectional	868	32,768
		S1-m	Unidirectional		
		S2	Bidirectional		
T	Frequent Transmit	T1	Unidirectional	868	100
		T2	Bidirectional		
R	Frequent Receive	R1	Unidirectional	868	4.8
		R2	Bidirectional		
N	Narrowband	N1	Unidirectional	169	-
		N2	Bidirectional		
C	Compact	C1	Unidirectional	868	50
		C2	Bidirectional		
F	Frequent Transmit and Receive			433	-

cation EN 13757-2 for using wired devices.

wM-Bus Communication Modes

wM-Bus supports various communication modes [58] [35] [59]. All modes are presented in Table 3.18

- **Stationary mode (S):** transmission between the meter and data collector in which the meter sends data a few times a day. In S1 mode, the data collector saves power before transmitting data. In S2, the transmitter requires an acknowledgment (ACK). S1-m is the same as S1, but the data collector is a mobile receiver.
- **Frequent transmit mode (T):** the meter devices send the data to collectors after a configurable duration, such as few transmission per second or minute. The transmitter requires an ACK after from the collectors.
- **Frequent receive mode (R):** the meter waits for a request from the collector before it transmits any data. Usually, the meter is in power saving mode and awakens over predefined intervals. R2 listens for a wake-up message from a transceiver on a regular basis. They have a few seconds to communicate after receiving the wake-up message.
- **Narrowband mode (N):** considered as multi-hop repeaters are used for long-range communication. It is designed for transmission in a low-frequency narrow band, and it is designed for long-range communication, including one-way, two-way, and forward communication.

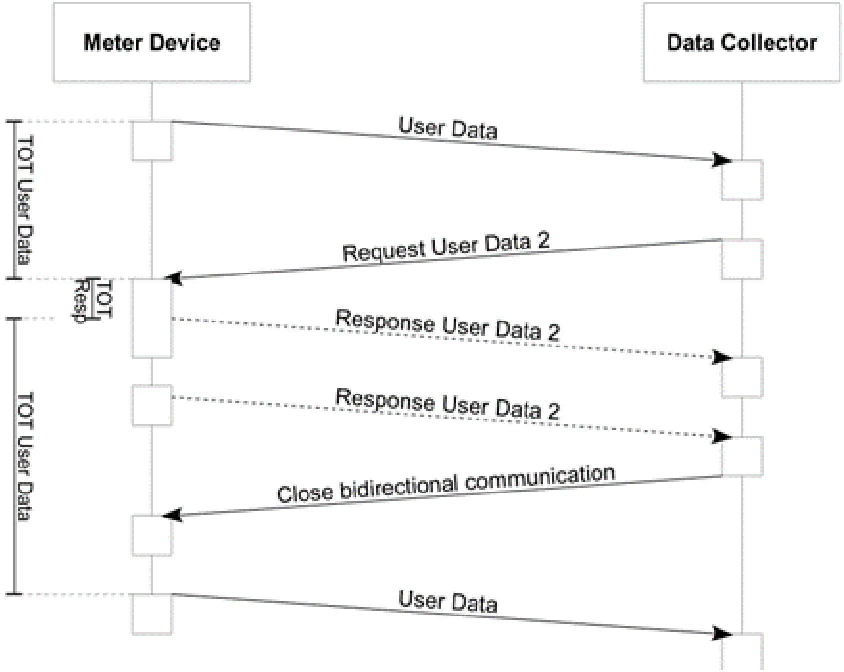


Figure 3.3: Bidirectional communication (Mode S2)

  - **N1:** one-way transmission, the node regularly transmits to a stationary receiving spot, allowing single-hop repeaters.
  - **N2:** two-way transmission, the node transmits same as N1. For a short period, the receiver stays active after each transmission then deactivate when the right preamble and synchronization state is detected.
- **Frequent transmit and receive mode (F):** is designed for long-range communication and is divided into one-way and two-way sub-modes.

wM-Bus consists of two communication modes: unidirectional and bidirectional. The unidirectional mode supports only data transmission from the meter to the data collector. Single devices implement this mode with low overhead, making it beneficial. The meter transmits data only, and the data collectors receive data only. Meanwhile, the bidirectional mode supports communication from the data collector to the metering device. Data collectors support bidirectional modes only, which can request data from bidirectional meter devices. The data collector sends a ”Request User Data 2” to the collector. The metering device receives the request and replies with a ”Response User Data 2” message containing information related to the data collector. The meter will repeat the response until the collector ends the communication or a timeout arises as presented in figure 3.3.

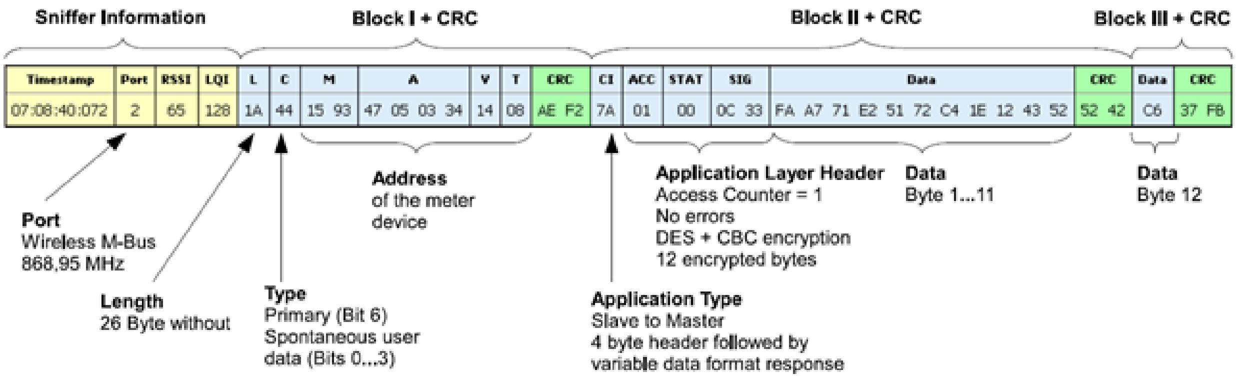


Figure 3.4: Example of a wM-Bus packet

Table 3.19: Wireless M-Bus Frame Format A [60]

First Block – Data Link Layer						
Length	Ctrl	ManID	Address	Ver	Type	CRC
1 byte	1 byte	2 bytes	4 bytes	1 byte	1 byte	2 bytes
Second Block – Application Layer						
CI	Data					CRC
1 byte	Up to 15 bytes					2 bytes
Optional Block – Application Layer						
Data						CRC
Up to 16 bytes						2 bytes

wM-Bus Data/Message Frame

A wM-Bus packet comprises of few blocks. It reserves 12 bytes for the first block, and the subsequent blocks hold 16 bytes. There is a checksum (16-bit) for each block transmitted, as shown in Figure 3.4. EN 13757-4 outlines two packet formats: format A and B, as illustrated in Table 3.19 & table 3.20. The multi-byte fields described in the following subsections transmit the least significant bytes first, except the CRC fields, which transmit the most significant bytes first.

wM-Bus Smart Grid Infrastructure

The wM-Bus infrastructure supports the 169, 433, and 868-MHz frequency bands, whereas the hardware maintains several operation modes:

- **Meter device:** an end-device collects and forwards data to other devices (Collector) such as gas and water meters.
- **Multi-Utility Controller / Gateway:** the component receives the data from the me-

Table 3.20: Wireless M-Bus Frame Format B [60]

<table>
<tr><th colspan="6">First Block – Data Link Layer</th></tr>
<tr><td>Length</td><td>Ctrl</td><td>Manuf.</td><td>Address</td><td>Ver</td><td>Type</td></tr>
<tr><td>1 byte</td><td>1 byte</td><td>2 byte</td><td>4 byte</td><td>1 byte</td><td>1 byte</td></tr>
<tr><th colspan="6">Second Block – Application Layer</th></tr>
<tr><td>CI</td><td colspan="2">Data</td><td colspan="3">CRC</td></tr>
<tr><td>1 byte</td><td colspan="2">Up to 115 bytes</td><td colspan="3">2 bytes</td></tr>
<tr><th colspan="6">Optional Block – Application Layer</th></tr>
<tr><td colspan="3">Data</td><td colspan="3">CRC</td></tr>
<tr><td colspan="3">Up to 126 bytes</td><td colspan="3">2 bytes</td></tr>
</table>

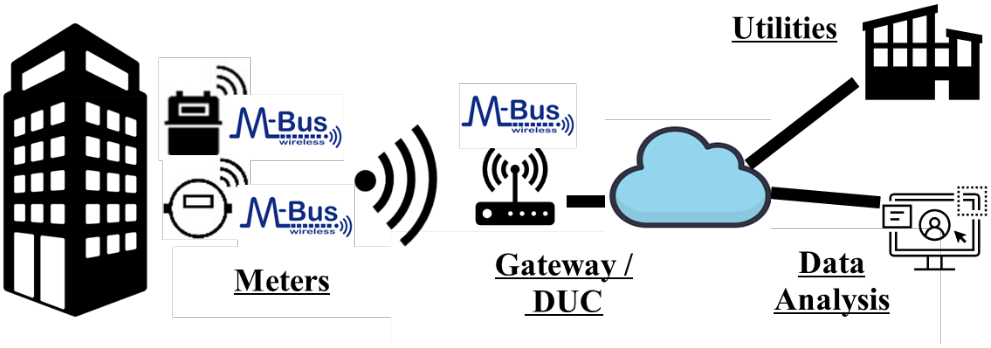


Figure 3.5: wM-Bus: Remote Metering Infrastructure

tering device, such as stationary receiver, mobile readout, or gateway.

- **Sniffer:** This component captures all communication during any transmission. It also captures decrypt encrypted communication.

A basic general wM-Bus architecture comprises of the sensor and gateway sides, as illustrated in Figure 3.5. The model architecture covers buildings in a neighbourhood, creating a 169/868-MHz wireless network in which the meter devices are considered the network sensor nodes. The communication between meters and the gateway/data collector is carried out via the wM-Bus protocol. Then, the communication between the gateway and the back-end services such as utility providers is carried out via the Internet.

wM-Bus Security

The focal weakness of wM-Bus lies not in the protocol or the standard specifications; instead, it is the implementation of the security, if there is any at all. Many property management/utility companies lean towards using unencrypted meter data or a single encryption key for all of their meters to save on effort during the deployment, decrease the cost of

Table 3.21: Wireless M-Bus Security Standards

Sec. Mode	**Algo**	**Mode of Operation**		**Security Services**			**OH byte**	**keys**	**Mac Size**
		Enc	**Auth**	**Conf**	**Intg**	**Auth**			
5	AES128	CBC	-	Y			2-17	1	0
7	AES128	CBC	CMAC	Y	Y	Y	2-23	2	2,4,8, 12, 16
8	AES128	CTR	CMAC	Y	Y	Y	6-16	3	4+2
9	AES128	GCM	GCM / GMAC	Y	Y	Y	18-21	1	12
10	AES128	CCM	CCM	Y	Y	Y	10-23	1	4,8, 12,16
13	TLS defined in OMS for Germany								

computation, and reduce the complexity of the security. Thus, it would make the meter vulnerable to attacks. L. Vegas [61] demonstrated comprehensive research on the wM-Bus's security issues. The general issues include a short key size, which is 64 bits, zero consumption detection, plain-text that shows information disclosure including the key, and man-in-the-middle attacks. However, the latest OMS specification introduced a security extension published by the OMS group [62], which allows for additional encryption modes 5, 7, 8, 9, and 10 as illustrated in Table IV while relying on AES-128 using dynamic keys, integrity preserving, and deriving the key from the authentication and fragmentation layer (AFL). In addition, mode 13 supports TLS 1.2 [63].

3.4.3 Proposed wM-Bus Security Framework

Overall Architecture

The proposed wM-Bus security framework is built based on OMS security standards. Besides, it should be a lightweight protocol to consider the meter's battery life. The framework will address the different layers of the wM-Bus Stack. The architecture of the proposed solution consists of two components Meter and the DUC/Gateway. should secure the communication between the local meter and the gateway within the local area network, which could be, for example, as a residential building. Each unit has a meter and one gateway to communicate with each meter to gather meter consumption information. The gateway can be at the side of the meters or the edge of the cloud.

Table 3.22: Wireless M-Bus Security Configuration

/! Initialization parameter for the decrypt and encryption.*
*Default: WMBUS_IV_METER_ADDR (As defined in EN13757) */*
#define WMBUS_IV_METER_ADDR 1
/! Initialization parameters for the decrypt and encryption.*
*Default: WMBUS_IV_METER_ADDR (As defined in EN13757) */*
#define WMBUS_IV_COLLECTOR_ADDR 2

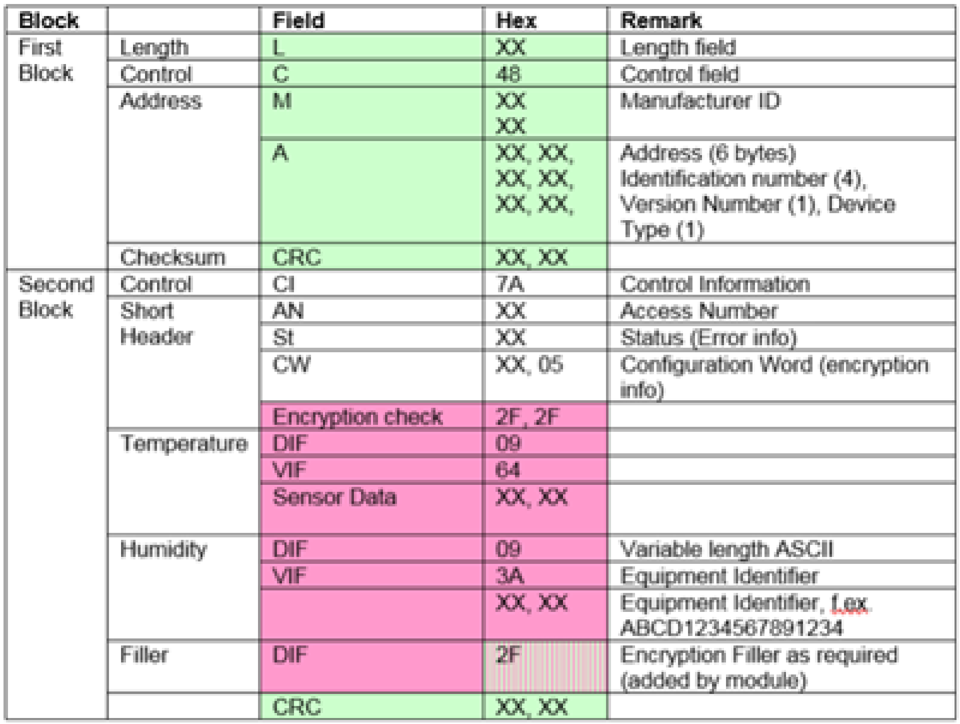
3.4.4 Security Framework base configuration

The main aims of the proposed security model are to ensure integrity and privacy/confidentiality. Integrity involves maintaining data accuracy to ensure data cannot be altered by unauthorized access during transmission. Meanwhile, privacy/confidentiality prevents unauthorized access to sensitive information. Integrity and privacy/confidentiality are often achieved through authentication, data encryption, and the use of dynamic keys (public and private). Therefore, to achieve all the security aspects within our proposed model, we need to make sure all security configurations are set at each side of the meter and gateway, as shown in Table 3.22. This is now possible due to the recent OMS standard v3 and v4, which have new security options. The default OMS version compiled with the stack is OMS v3 configured by defining macro @ref OMS_ENABLED. The following are the significant new features of OMS v4:

- The AFL splits long messages up to 16 KB into several fragments as is the wM-Bus telegrams standard. It calculates a block cipher-based message authentication code (CMAC) algorithm.
- Encryption Mode-7 at the transport layer (TPL) allows dynamic keys to be used for the encryption and decryption of a message.

The Proposed Security Framework Aspects

OMS defines several security profiles currently using profile A mode 5, B mode 7, and C mode 13 (specific for Germany) as presented in Table 3.4.2. As for our security framework, we will define the new profile called ‘W’. This security profile comprises Mode 9: AES-128-GCM, authentication is GCM, GMAC with 12 Byte MAC in AFL, and asymmetric key. The reason for adopting mode 9 is that GCM is considered superior to CCM for most applications that require authenticated encryption. Moreover, message authentication GMAC/GHASH is done on the ciphertext. Additionally, authentication verification and decryption occur in parallel for performance reasons in most implementations.



Block		Field	Hex	Remark
First Block	Length	L	XX	Length field
	Control	C	48	Control field
	Address	M	XX XX	Manufacturer ID
		A	XX, XX, XX, XX, XX, XX,	Address (6 bytes) Identification number (4), Version Number (1), Device Type (1)
	Checksum	CRC	XX, XX	
Second Block	Control	CI	7A	Control Information
	Short Header	AN	XX	Access Number
		St	XX	Status (Error info)
		CW	XX, 05	Configuration Word (encryption info)
		Encryption check	2F, 2F	
	Temperature	DIF	09	
		VIF	64	
		Sensor Data	XX, XX	
	Humidity	DIF	09	Variable length ASCII
		VIF	3A	Equipment Identifier
			XX, XX	Equipment Identifier, f.ex. ABCD1234567891234
	Filler	DIF	2F	Encryption Filler as required (added by module)
		CRC	XX, XX	

Figure 3.6: Example of a wM-Bus message that includes an encryption configuration and other data added by the model [6]

Based on the communication mode set between the meter and the gateway, the data will be transmitted through one-way communication (unidirectional) such as S1, T1, and N1 modes or two-way (bidirectional) S2, T2, and N2 modes. Once the initial communication starts, the meter will initiate the wake-up state based on the preconfigured interval. The meter will send the parameters required to communicate securely with the gateway. During the initial communication, both need to have secure transmission. They do not need to agree on the cipher suite or encryption/decryption algorithm since it supports advanced encryption standards (AES-128). Each message may be encrypted using an additional shared secret key and asymmetric encryption algorithm. The one overhead over the meter side calculates the master session key and performs mutual authentication with the gateway. During the key exchange, mutual authentication can be performed to ensure the proposed framework is lightweight. We could overcome the overhead on the meter by increasing the reading intervals. For example, some utility companies set up the reading from the meter once every 30 days.

To build a lightweight protocol, we need to reduce the cost of power on the meter side. Since the meter and gateway are configured to the security suite, the negotiation of the security parameters will be eliminated. The handshake will only consist of two main tasks:

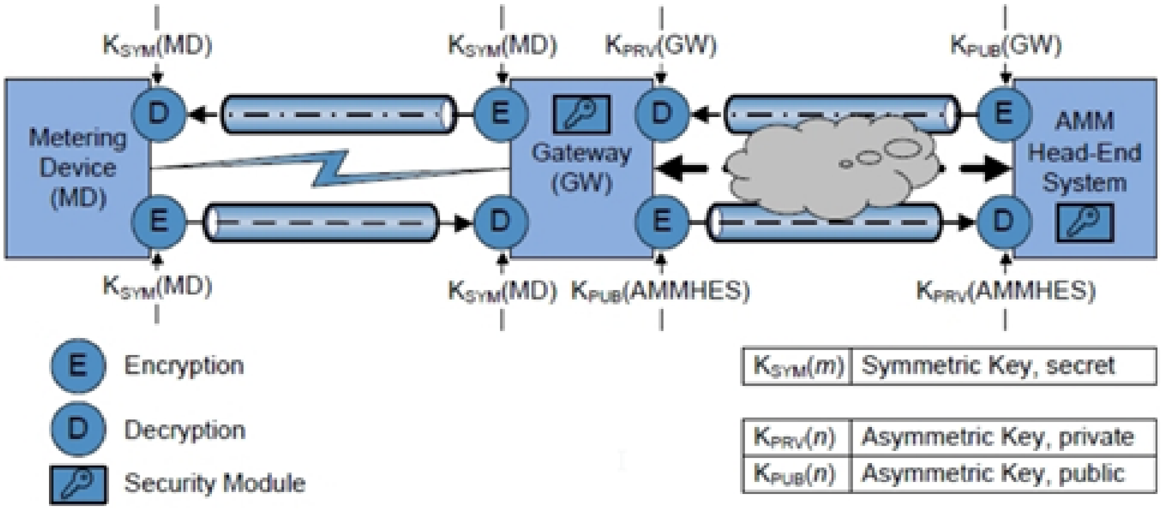


Figure 3.7: wM-Bus: Remote Metering Infrastructure

establishing a secret key for symmetric encryption and authentication by asymmetric key and sharing the secret key using Elliptic-Curve Diffie-Hellman Ephemeral (ECDHE) cryptography. The process will transmit the encrypted messages, as when it is received by the gateway it can decrypt with the symmetric secret key and verify the message integrity. The message will include unencrypted data that the meter model adds, such as the manufacturer ID, address, and identification number, as shown in Figure 3.6. This associate (added) data will go to the GMAC algorithm as part of the encryption and decryption process called authenticated encryption with associated data (AEAD). As for the key derivation function (KDF), the encryption relies on the ephemeral key, which only uses one message. KDF is based on CMAC. In our proposal, we will consider the assumption of the pre-shared key and allow the meter and the gateway to derive their secret key for encryption and authentication. There should be a unique key for each session and a unique key for message encryption. An example of encryption and decryption is illustrated in Figure 3.7.

Summary

The proposed framework suggested implementing a new security profile while maintaining a lightweight protocol to save on the meter battery life. The proposed framework considers the security factors such as integrity and confidentiality/privacy. This section summarized the main points of our proposed framework.

- Meters and gateways with the wM-Bus protocol should be configured to enable security functions.

- Security is defined by the TPL, as it determines which security mode is used.
- AFL should be used to support authentication, as it helps in maintaining the message frame length on a short frame data channel [63].
- Encryption using AES-128 bit, with CBC and CTR modes.
- Authentication is done by adding MAC hashed message authentication codes (HMACs) with the GMAC algorithm.
- The keys derivation function should be used, which means we never use the same key twice.
- The message counter derives a unique counter value for the encryption's initial vector (IV) or for making a unique key for the key derivation function.
- There are different security modes such as 5, 7, 8, 9, 10 and 13. Mode 9 has been proposed for our framework.
- Part of the negotiation should be eliminated during the handshake to maintain the meter cost with power consumption. This is due to the security pre-configuration.

Chapter 4

4 Lightweight Protocol Over wM-Bus for IoT Echosystems

4.1 Protocol Overview

We are going to propose a new model architecture, in which the incorporation of the wM-Bus protocol, specifically through the employment of the Collector N Mode, serves as the fundamental conduit for data exchange between the Meter and the Gateway. This framework is further strengthened by the adoption of the NPF, enhancing both the efficiency and security of data transmission against potential threats. As illustrated in Figure 4.1, as our case shows the meter side with the water sensor employs a Radiocraft RF module, operating at sub-1 GHz frequencies, notably at 169Hz in the N Mode, emphasizing its capacity for extensive range and penetration, a critical attribute for utility metering across diverse settings. This advanced integration forms the backbone of our secure and efficient communication system. As for the efficiency the model compliance with the wM-Bus protocol's packet frame specifications. As illustrated in Figure 4.2, the design leverages the 10-byte header to encapsulate essential wM-Bus information alongside meter device details. For a more detailed exploration of the header implementation, please refer to Chapter 5, where comprehensive insights and technical specifics are thoroughly discussed. Following the header, the protocol dedicates a subsequent segment specifically for facilitating message exchanges within the NPF, thereby enabling secure handshake communications.

4.2 Protocol Communications

The meter, assuming the role of Initiator, and the gateway, acting as the Responder, are both integrated with their respective wM-Bus RF models and the NPF. This setup underscores the dual emphasis on adherence to established protocol specifications for wireless communication and the enhancement of security protocols via the NPF. This integration ensures efficient and secure data flow, adhering to wM-Bus's packet frame specifications, thus meeting both security and compatibility standards essential for smart metering applications within IoT environments.

Our proposed model of handshake communication, integrating wM-Bus protocols with

---

[0] A version of this chapter has been accepted for publication in IEEE Canadian Journal for Electrical and Computer Engineering (CJECE) March 2024

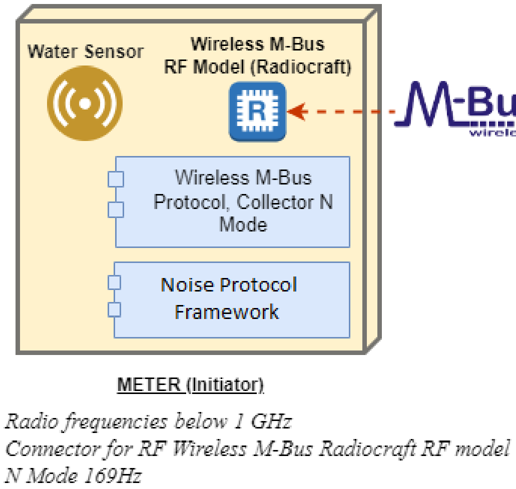
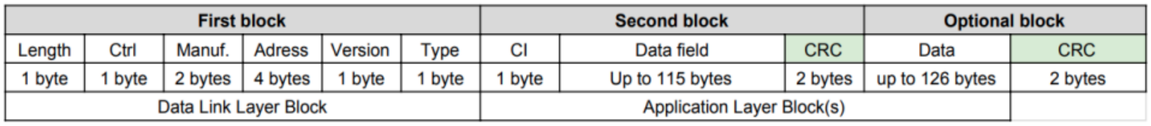


Figure 4.1: Overview of the proposed model: communication setup demonstrating the integration of the wM-Bus protocol with NPF for secure data transmission



First block						Second block			Optional block	
Length	Ctrl	Manuf.	Adress	Version	Type	CI	Data field	CRC	Data	CRC
1 byte	1 byte	2 bytes	4 bytes	1 byte	1 byte	1 byte	Up to 115 bytes	2 bytes	up to 126 bytes	2 bytes
Data Link Layer Block						Application Layer Block(s)				

Figure 4.2: wM-Bus Packet Frame

the NPF, clearly outlines the communication process through a sequence of distinct steps, demonstrating the secure exchange of data:

1. Pre-Handshake Preparation: All devices are ensured to support the NPF in addition to the wM-Bus protocol. Each device generates a static key pair for use in the NPF XX handshake.
2. Handshake Initiation: The initiator generates an ephemeral key pair and sends the public key to the responder.
3. Handshake Response: Upon receiving the ephemeral public key, the responder generates its own ephemeral key pair. The responder encrypts its ephemeral public key using the initiator's ephemeral public key and sends it back, along with its encrypted static public key.
4. Handshake Completion: The initiator decrypts the responder's ephemeral and static public keys. It then encrypts its static public key using the responder's ephemeral public key and sends it to the responder. Both parties now have shared their ephemeral and static public keys, encrypted with the derived shared secrets.

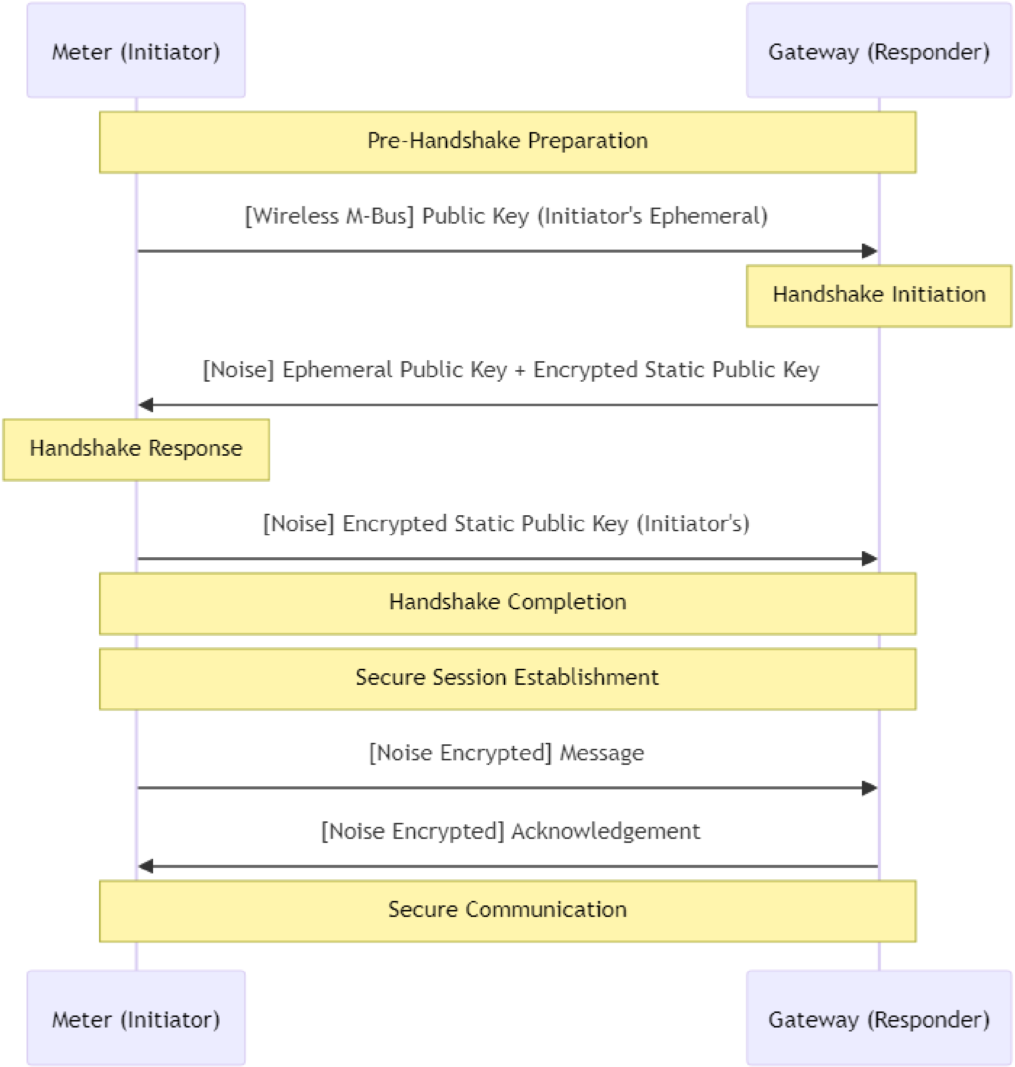

Meter (Initiator)
Gateway (Responder)
Pre-Handshake Preparation
[Wireless M-Bus] Public Key (Initiator's Ephemeral)
Handshake Initiation
[Noise] Ephemeral Public Key + Encrypted Static Public Key
Handshake Response
[Noise] Encrypted Static Public Key (Initiator's)
Handshake Completion
Secure Session Establishment
[Noise Encrypted] Message
[Noise Encrypted] Acknowledgement
Secure Communication
Meter (Initiator)
Gateway (Responder)

Figure 4.3: Secure wM-Bus Communication Handshake Using the NPF with XX Pattern Between Initiator (Meter) and Responder (Gateway)

5. Secure Session Establishment: Both parties use the NPF functions to derive a shared secret from the exchanged keys. This shared secret is used to encrypt and decrypt subsequent communications.
6. Secure Communication: The established shared secret facilitates the encryption and decryption of messages in alignment with the NPF's symmetric encryption schemes. This ensures that messages are securely transmitted in accordance with the wM-Bus protocol specifications, with the added layer of encryption enhancing the security of the wM-Bus protocol.

By integrating the NPF's XX pattern, the wM-Bus protocol can achieve a higher level of security, ensuring the confidentiality, integrity, and authenticity of the communications within smart metering applications.

4.3 Protocol Assumptions

Our assumptions about device architectures and trust models also shape the limitations of this research. Each device's identity, type, and version, along with unique keys for meters and gateways, and the support for the NPF, underpin the protocol's security mechanisms. However, these assumptions may not hold in all potential application scenarios, limiting the applicability of our findings. Moreover, the trust assumptions related to the Diffie-Hellman key exchange and the possibility of employing Trust on First Use (TOFU) in environments lacking infrastructure for digital certificates or pre-shared keys represent a significant consideration. While these assumptions are suitable for the context of wM-Bus, they may not be universally applicable or acceptable, particularly in scenarios with higher security requirements or where the initial communication may be at risk of interception. Additionally, concerns about the interoperability of the wM-Bus protocol with the NPF must be addressed. The wM-Bus protocol comes in several versions (e.g., S, T, C, N), each with unique specifications and requirements. The NPF must be tailored to work with the specific version of the wM-Bus protocol in use. The selection of the NPF pattern (e.g., XX, NK, NX, IK) should be based on the wM-Bus communication's specific requirements, factoring in the distinct features and capabilities of each pattern. Performance considerations are paramount, as the NPF must operate within the limited processing power, memory, and battery life of wM-Bus devices, necessitating optimization for low power consumption and efficient resource use.

4.4 Protocol Design Methodology

The research presents a structured approach to protocol design, focusing on the practical implementation and evaluation of communication protocols within secure environments. This is specifically applied to the NPF and the wM-Bus protocol in IoT settings. The protocol design methodology is structured into distinct phases:

1. Framework Implementation and Preliminary Analysis: This phase establishes benchmarks for NPF performance and provides a detailed analysis of the wM-Bus protocol, focusing on communication duration, packet size, and memory usage. It eval-

uates NPF's functionality with various security configurations, assessing the effects of encryption ciphers, key exchange methods, and hash functions. Simultaneously, it assesses the wM-Bus protocol's efficiency in IoT devices in simulated metering scenarios.

2. Simulation and Integration: This phase upgrades the wM-Bus protocol with TLS 1.2 for better security and evaluates its performance against standard metrics. It also dynamically analyzes the NPF's 12 patterns to identify those that optimize security, packet size, and handshake duration, aiming to understand the impact of various security patterns on key performance indicators.
3. Comprehensive Testing and Optimization: The methodology concludes with an exhaustive assessment of the NPF, integrating all security properties to propose an optimized, lightweight protocol. This phase synthesizes insights from all preceding stages, offering a holistic view of the performance characteristics, strengths, and weaknesses observed, leading to the proposition of a refined, secured protocol.

The research focuses on IoT devices in wM-Bus networks, using tools like the Radiocraft developer kit [64] for data collection on memory usage, packet size, handshake time, and security features. It uses Noise Explorer to check NPF's security and Python for integration throughout the study. The research evaluates security with the STRIDE model and efficiency and battery impact, noting that the evaluation found security measures reduce battery life as a trade-off compared to non-secured operations. To ensure a thorough evaluation, our study utilized a carefully designed implementation setup, crucial for assessing the effectiveness and relevance of the selected NPF patterns. The setup Involved the assembly of the hardware and software components. This setup aimed to authentically replicate the interaction between a Gateway (Server) and a Meter (Client) within a wireless communication context.

4.5 Protocol Implementation Setup

The hardware foundation was established using two Radiocrafts developer kit as Figure 4.4 shows, wireless communication cards, model RC1701HP–MBUS4 from the RC17xx series, to simulate the pivotal roles of Gateway and Meter. These cards, selected for their robust support for N-mode operation at a 169 MHz frequency, are available in two power variants: High Power (27dBm) and Very High Power (30dBm), catering to diverse operational demands. Complementing the wireless modules, a high-performance laptop equipped with an Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz and 12.0 GB of RAM under a 64-

bit architecture was employed. This computational device was instrumental, providing the necessary processing power and interface for the simulation of the Server and Client roles, as illustrated in Figure 4.5.

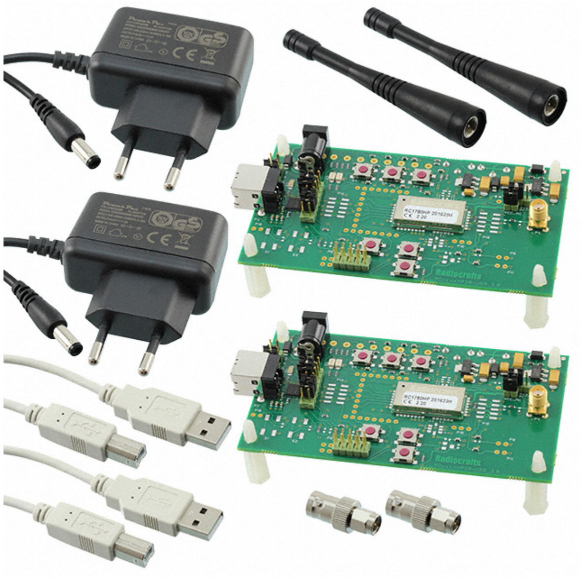


Figure 4.4: wM-Bus development kit RC1701HP-MBUS4-DK

On the software part, the RCTools suite emerged as a cornerstone for the initial configuration and diagnostic assessment of the wM-Bus Cards communication. This suite, specifically tailored for the wM-Bus family, incorporates tools like CCT and Demo, streamlining the development and deployment phases for Radiocrafts modules. The installation process, characterized by its simplicity due to the integration of .NET and USB Driver components, facilitates a seamless setup experience. Concurrently, a Python-based development environment was architected, focusing on the nuanced dynamics of the wM-Bus protocol. This environment leveraged specialized wM-Bus libraries alongside tools such as “tracemalloc” for memory management and “PySerial” for serial communications, ensuring efficient dialogue between the cards. Configurations were established using “COM3” and “COM4” ports for the cards, with a baud rate set at 12900, optimizing the connection setup and data exchange process. Notably, the decision to transmit data in plaintext, devoid of any security measures, was made to establish a foundational baseline for the assessment of wM-Bus communications, laying the groundwork for subsequent explorations into security enhancements.

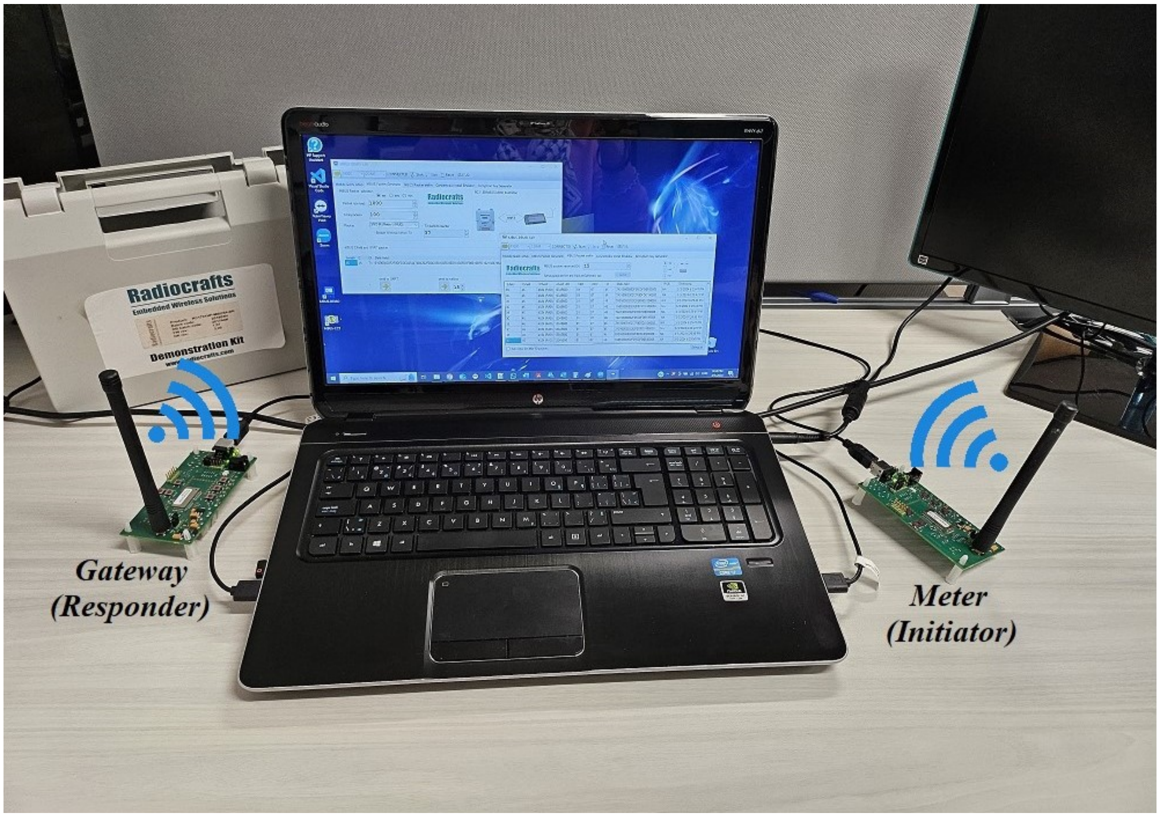


Figure 4.5: A wM-Bus communication setup with a laptop connected to a Gateway (Responder), and a Meter (Initiator), part of a Radiocrafts demonstration kit.

Chapter 5

5 Implementation of Protocols to Secure Wireless Communication

5.1 Introduction

This chapter presents a thorough examination of the NPF and TLS protocols within the context of securing wM-Bus communications, focusing on critical performance metrics such as memory usage, packet size, and handshake process time. Spanning five distinct phases, the analysis begins with baseline performance metrics of the NPF, followed by a deep dive into its predefined patterns. Subsequent phases extend the investigation to the integration of the NPF with wM-Bus, optimization strategies for lightweight protocols, and a conclusive comparison with TLS to ascertain the most efficient and secure method for wireless metering communications. This multifaceted approach not only evaluates individual and combined protocol efficiencies but also contrasts them against the established TLS standard, aiming to delineate a clear pathway towards optimizing security and performance in battery-powered metering devices.

The structure of this chapter is outlined in the following phases and is also visually represented in Figure 5.1:

1. **Phase One - NPF Baseline Performance Metrics:** The purpose of the first phase, is to establish a foundational understanding of the NPF's performance characteristics.
    - **Phase Activities and setup:** This phase involved a detailed analysis of the NPF patterns, each with specific security options in terms of key exchange, authentication mechanism, encryption method, and hashing algorithm. The Key performance metrics are time taken for communication, packet size, and memory consumption were measured for the NPF operating solo on server and client terminals.
    - **Outcome:** The outcome of this phase was a comprehensive baseline of performance metrics for the NPF, providing a clear picture of how each pattern performs in isolation. This baseline served as a crucial reference point for subsequent phases of the project, where these patterns were further tested and optimized in combination with the wM-Bus protocol and compared against other security protocols like TLS.

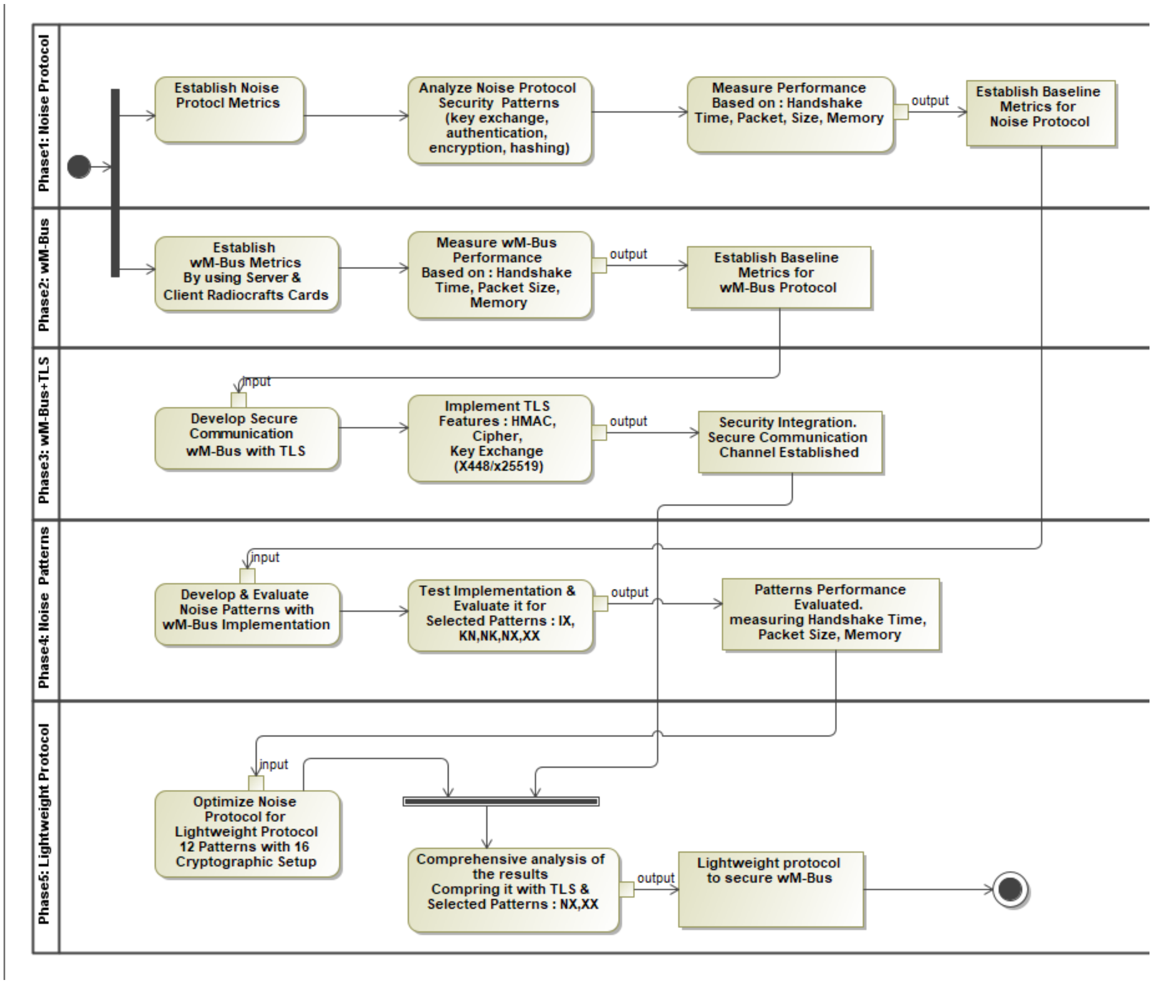

Phase1: Noise Protocol
Establish Noise Protocl Metrics
Analyze Noise Protocol Security Patterns (key exchange, authentication, encryption, hashing)
Measure Performance Based on : Handshake Time, Packet, Size, Memory
output
Establish Baseline Metrics for Noise Protocol
Phase2: wM-Bus
Establish wM-Bus Metrics By using Server & Client Radiocrafts Cards
Measure wM-Bus Performance Based on : Handshake Time, Packet Size, Memory
output
Establish Baseline Metrics for wM-Bus Protocol
Phase3: wM-Bus+TLS
input
Develop Secure Communication wM-Bus with TLS
Implement TLS Features : HMAC, Cipher, Key Exchange (X448/x25519)
output
Security Integration. Secure Communication Channel Established
Phase4: Noise Patterns
input
Develop & Evaluate Noise Patterns with wM-Bus Implementation
Test Implementation & Evaluate it for Selected Patterns : IX, KN,NK,NX,XX
output
Patterns Performance Evaluated. measuring Handshake Time, Packet Size, Memory
Phase5: Lightweight Protocol
input
Optimize Noise Protocol for Lightweight Protocol 12 Patterns with 16 Cryptographic Setup
Comprehensive analysis of the results Compring it with TLS & Selected Patterns : NX,XX
output
Lightweight protocol to secure wM-Bus

Figure 5.1: Activity Diagram for Implementation and Evaluation Phases

2. **Phase Two – wM-Bus Protocol Baseline Performance Metrics:** The purpose of the second phase, was to establish a foundational benchmark for the performance of the wM-Bus protocol in isolation. This phase was crucial for understanding the inherent capabilities and limitations of the wM-Bus protocol before integrating it with the NPF. The key activities and outcome of this phase:

   - **Setup:** The experimental setup involved running the wM-Bus protocol independently on two Radiocrafts cards configured as client and server. This setup was crucial for evaluating the protocol's performance in a controlled environment, closely mimicking real-world metering applications. Same key metrices used; time taken for the handshake communication between the client and server, the size of the packets transmitted, and the overall memory consumption during the handshake.

  - **Outcome:** The outcome of this phase was a comprehensive set of baseline performance metrics for the wM-Bus protocol. These metrics served as a reference point for subsequent phases of the project, particularly when evaluating the impact of integrating the NPF and assessing the performance trade-offs involved in securing wM-Bus communications.

3. **Phase Three - wM-Bus with TLS 1.2:** The third phase, was dedicated to enhancing the security of the wM-Bus communication protocol by incorporating TLS (Transport Layer Security) layer. This phase built upon the initial development of Python code for wM-Bus communication, which previously lacked security measures. The primary goals and outcomes of this phase included:

   - **TLS Implementation:** utilizing key Python libraries such as socket and cryptography, this phase implemented essential cryptographic functionalities like HMAC (Hash-based Message Authentication Code), cipher operations for encryption and decryption, hashing, and HKDF (HMAC-based Extract-and-Expand Key Derivation Function). These tools were instrumental in establishing a secure communication channel between the wM-Bus client (meter) and server (gateway). Furthermore, a comprehensive approach was adopted by integrating various key exchange mechanisms (such as X448 and X25519) with each cipher suite to thoroughly examine their security properties.

   - **Outcome and Analysis:** The phase meticulously recorded and analyzed the outcomes of implementing a TLS secure layer within the wM-Bus communication protocol. Execution logs for each cipher suite provided detailed information on the handshake process, including steps like ’Client Hello’, ’Server Hello’, ’Client Finish’, and ’Server Finish’, along with the transmission of application data. These logs were crucial for evaluating the security properties, memory usage, and elapsed time for each phase of the communication, allowing for a granular examination of each cipher suite’s impact on the security and performance of the wM-Bus communication.

4. **Phase Four - NPF with Selected Patterns:** The focus was on integrating the NPF with wM-Bus technology to enhance the security of communications between meters and gateways. This phase can be summarized in three main aspects:

   - **Setup and Activities:** Developing wM-Bus technology, seamlessly integrating it with the NPF. This involved implementing and evaluating six distinct predefined communication patterns: IX, KN, NK, NX, and XX, within the context of the NPF. Python was used to develop code that utilized both the wireless and NPF libraries, with a primary objective to prioritize and enhance the security of communication between the meter and the gateway.

  - **Implementation:** This implementation were rigorously tested using consistent metrics throughout the process, focusing on memory usage and the time required for the handshake process. This was meticulously documented for the gateway and meter, as well as the entire handshake and encrypted application data transmission process.
  - **Outcome:** The outcome of this phase was a detailed evaluation of the performance and resource demands of the integrated system, aiding in fine-tuning and optimizing for real-world deployment. Specifically, the analysis revealed that the XX pattern had significantly lower total memory usage compared to other patterns, making it potentially more suitable for devices with limited memory resources. In terms of time efficiency, the KN pattern was found to be the most time-efficient, with the shortest total time required for the handshake process, while the XX pattern, despite its low memory usage, took the longest total time, possibly due to additional steps involved in its handshake process.

5. **Phase Five - Optimizing NPF for Lightweight Protocol:** This phase aims at optimizing the NPF toward lightweight protocol, running 12 patterns and incorporating all security properties to propose the best-suited protocol for battery-powered meters.This phase can be broken down into three main components:
   - **Setup and Activities:** The investigation maintained a consistent hardware setup using Radiocrafts cards, specifically the ”RC1701HP-WMBUS4” model, to accurately simulate the functionality of both meter and gateway.
   - **Implementation:** The methodology incorporated specialized Python libraries for wM-Bus communication and the NPF. Cryptographic elements like AES-GCM, ChaCha20-Poly1305, X25519, and X448 curves, along with SHA-256, SHA-512, Blake2s, and Blake2b hashing algorithms, were selected for their efficiency and security.
   - **Outcome:** Throughout the experiment, 12 distinct patterns paired with approximately 16 security configurations were meticulously executed to evaluate their performance and compatibility. However, two patterns were deliberately omitted due to their oversized packet lengths exceeding the wM-Bus protocol’s maximum allowable dimensions. Each execution cycle was methodically recorded,

with outcomes systematically documented, ensuring a detailed and accessible record of the results. This facilitated an in-depth analysis and comparison of the various security configurations tested, providing a comprehensive overview of the experiment's findings and highlighting the security implications of each tested pattern and configuration.

6. **Summary of the Comparative Analysis and Final Comparison** in this two section, the focus was on evaluating different NPF patterns and their suitability for securing wM-Bus communications, culminating in a comparison between the NPF and TLS protocols. The analysis aimed to identify the most efficient and secure method for wireless metering communications, considering factors like memory usage, packet size, and handshake times. The final comparison highlighted the XX pattern within the NPF as particularly suitable for IoT devices like smart meters due to its balance of security and resource efficiency.

    - **Pattern Comparative Analysis:** The analysis involved comparing different NPF patterns, focusing on their memory usage, packet size, and handshake times. The XX pattern for the three messages category, and NX pattern for the two messages category, emerged as particularly efficient, offering low memory usage and smaller packet sizes, which are crucial for battery-powered IoT devices.

    - **Comparison with TLS:**The research compared the performance and security of wM-Bus communications secured with TLS and the NPF. While TLS is a widely used security protocol, the comparison aimed to assess its suitability against the NPF for the specific needs of wM-Bus communications.

    - **Outcome:** The XX pattern within the NPF was identified as a highly suitable option for securing wM-Bus communications. It provides an effective balance between security and resource efficiency, making it ideal for battery-powered IoT devices. The aesgcm-x2551-blake2b configuration, in particular, showed the least memory usage, while aesgcm-x2551-blake2s suggested better bandwidth efficiency. The handshake and transmission times were relatively consistent across all combinations, indicating stable performance.

Table 5.1: Noise Framework Predefined Patterns

Pattern	**Security**	**Key Exchange**	**Authentication**	**Encryption**	**Hashing**	Rnak
XN	Low	X25519	None	None	None	1
KK	Low	Symmetric (Pre-shared keys)	None	ChaCha20-Poly1305 AEAD	None	
NN	Low	None (Symmetric keys derived from pre-shared key)	HMAC-SHA257	None (Unencrypted)	SHA256	
IN	Moderate	X25519	HMAC-SHA256	None	SHA256	2
XK	Moderate	X25519	None	ChaCha20-Poly1305 AEAD	BLAKE2s	
NK	Moderate	X25519	HMAC-SHA256	None	SHA256	
IK	High	X25519	None	ChaCha20-Poly1305 AEAD	BLAKE2s	3
KX	High	X25519	None	ChaCha20-Poly1305 AEAD	BLAKE2s	
XX	High	X25519	None	ChaCha20-Poly1305 AEAD	BLAKE2s	
NX	Very High	X25519	HMAC-SHA256	ChaCha20-Poly1305 AEAD	SHA256	4
IX	Very High	X25519	None	ChaCha20-Poly1305 AEAD	BLAKE2s	
KN	Very High	X25519	HMAC-SHA256	ChaCha20-Poly1305 AEAD	SHA256	

5.2 Phase One: NPF

NPF was deployed in isolation to establish fundamental performance benchmarks crucial for subsequent analysis. This phase, pivotal in laying the groundwork for the entire study, revolved around the meticulous selection and execution of 12 distinct NPF patterns, as shown in Table 5.1. Each pattern was chosen for its unique combination of security features, including key exchange mechanisms, encryption methods, and hashing algorithms, thereby offering a diverse spectrum of security configurations.

The primary metrics evaluated during this phase were the time taken for communication, packet sizes, and memory consumption. This phase was not merely about data collection but also about understanding the intricate balance between security and efficiency that these patterns represented.

By rigorously analyzing these patterns in a controlled environment, the research aimed to discern the inherent trade-offs and synergies each pattern offered. This foundational analysis was instrumental in setting the stage for subsequent phases, where these patterns were further tested in conjunction with the wM-Bus protocol, thus enriching the research with empirical data and insights that would guide the development of an optimized security protocol for low-power communication systems.

5.2.1 NPF Predefined Pattern Analysis

This section thoroughly examines a variety of patterns, concluding with a detailed summary in Table 5.2. Each pattern is carefully analyzed to assess its impact and efficiency within the framework. The evaluation is organized to provide insights into how effective and practical each pattern is in relation to the objectives of the research. The summary table acts as a crucial reference, presenting the main outcomes from the in-depth analysis of the patterns, thereby offering a clear overview of their advantages, limitations, and potential roles in improving the protocol's security and efficiency.

XN Pattern

The lack of authentication, encryption, and the use of X25519 for key exchange in wireless communication between a meter and a gateway via wM-Bus raises significant security concerns.

- No Authentication: Without authentication, there is no way to ensure the identity of the communicating parties. This opens the door for unauthorized devices to potentially join the network, impersonate legitimate devices, or intercept and manipulate data.
- No Encryption: Lack of encryption means that the data transmitted between the meter and gateway is vulnerable to eavesdropping and tampering. Attackers could intercept sensitive information, manipulate readings, or inject malicious data into the communication stream without detection.
- Use of X25519 for Key Exchange: X25519 is a widely accepted elliptic curve Diffie-Hellman (ECDH) key exchange algorithm. While it's a strong choice for establishing a shared secret between the meter and gateway, it doesn't address the lack of authentication and encryption in overall communication.
- Security Implications: The absence of authentication allows for the potential compromise of the entire communication system. Unauthorized access to the network could lead to unauthorized data access, disruption of service, or injection of false data. Lack of encryption exposes sensitive data, such as meter readings and customer information, to unauthorized access or manipulation. This is a serious privacy concern and could lead to financial or reputational damage.

KK Pattern

In KK pattern the key exchange is done using a symmetric pre-shared key (PSK) in a pattern like KK, with no authentication, encryption using ChaCha20-Poly1305 AEAD, and no hashing raises some specific security considerations:

- Symmetric Pre-shared Key (PSK) for Key Exchange: The use of a symmetric pre-shared key for key exchange is a reasonable approach when there is a need for simplicity and efficiency. However, security relies heavily on the protection and distribution of this shared key. Any compromise of the pre-shared key could lead to a complete breakdown of security.

- No Authentication: The absence of authentication means there is no mutual verification of the communicating parties. Without proper authentication, it becomes possible for unauthorized entities to gain access to the network, perform man-in-the-middle attacks, or impersonate legitimate devices.
- Encryption Using ChaCha20-Poly1305 AEAD: ChaCha20-Poly1305 is a modern symmetric encryption algorithm with authenticated encryption with associated data (AEAD). It provides both confidentiality and integrity, protecting the data from eavesdropping and tampering. This is a strong choice for securing the communication channel.
- No Hashing: The absence of hashing could be a concern, especially if there's a need for data integrity verification. Hashing is commonly used to ensure that the received data has not been altered during transit. Without hashing, there is no built-in mechanism to verify the integrity of the exchanged messages.
- Security Implications: While the use of ChaCha20-Poly1305 AEAD provides strong encryption, the lack of authentication poses a significant risk. An attacker might be able to gain unauthorized access, inject malicious data, or manipulate the communication without detection. The reliance on a pre-shared key for key exchange requires careful key management. Any compromise of the key could have severe consequences.

NN Pattern

Analyzing the security properties of the NN pattern with the given specifications:

- Key Exchange: Symmetric Keys Derived from Pre-shared Key (PSK): In this pattern, symmetric keys are derived from a pre-shared key (PSK). This approach simplifies the key exchange process, but the security heavily relies on the protection and distribution of the pre-shared key. If the PSK is compromised, the entire security of the system is at risk.
- Authentication: HMAC-SHA257: The use of HMAC-SHA257 for authentication is a strong choice. HMAC (Hash-based Message Authentication Code) provides integrity and authenticity, and SHA-257 is a secure hash function. The combination ensures

that the received data has not been tampered with and comes from a legitimate source.

- No Encryption: The absence of encryption means that the data is transmitted in plaintext. This exposes the information to eavesdropping, and attackers could potentially gain access to sensitive data. It's important to consider the sensitivity of the transmitted information and whether encryption is necessary to protect confidentiality.
- Hashing: SHA256: SHA256 is a widely used and secure hash function. Its purpose in this context is likely for integrity verification. Hashing ensures that the data has not been altered during transmission. However, it's crucial to note that hashing alone does not provide confidentiality.
- Security Implications: The reliance on a pre-shared key for key derivation means that the security of the system is as strong as the protection of that key. Adequate key management practices, including secure distribution and periodic key updates, are essential. The use of HMAC-SHA257 provides strong authentication, but the lack of encryption means that the data is exposed during transit. This may be acceptable for certain use cases where confidentiality is not a primary concern. The choice of SHA256 for hashing contributes to data integrity, but it's important to ensure that the hash is appropriately validated on both ends to detect any tampering.

IN Pattern

Analyzing the security properties of the IN pattern with the given specifications:

- Key Exchange: X25519 is a modern and widely accepted elliptic curve Diffie-Hellman (ECDH) key exchange algorithm. It provides a secure way for two parties to agree on a shared secret over an insecure communication channel. The use of X25519 is a strong choice for key exchange.
- Authentication: The use of HMAC-SHA256 for authentication is a robust choice. HMAC provides integrity and authenticity, and SHA256 is a secure hash function. This combination ensures that the exchanged data has not been tampered with and comes from a legitimate source.
- No Encryption: The absence of encryption means that the data is transmitted in plaintext, exposing it to potential eavesdropping. Depending on the nature of the transmitted information, the lack of encryption may or may not be a concern. It's important to consider the sensitivity of the data being exchanged.
- Hashing: SHA256 is a widely used and secure hash function. Its role in this context

is likely for integrity verification. Hashing ensures that the data has not been altered during transmission.

- Security Implications:The use of X25519 for key exchange provides a strong foundation for secure communication, as it helps establish a shared secret between the communicating parties. HMAC-SHA256 contributes to authentication, ensuring that the data is not tampered with and comes from a legitimate source. The lack of encryption means that the data is exposed during transmission. This may be acceptable for certain use cases where confidentiality is not a primary concern, but it's crucial to assess the specific security requirements of the application.

XK Pattern

Analyzing the security properties of the XK pattern with the given specifications:

- Key Exchange: X25519 is a modern and secure elliptic curve DiffieHellman (ECDH) key exchange algorithm. It enables the two parties to agree on a shared secret over an insecure communication channel. The use of X25519 for key exchange is a strong choice.
- No Authentication: The absence of authentication means there is no mutual verification of the communicating parties. Without proper authentication, it becomes possible for unauthorized entities to gain access to the network, perform man-in-the-middle attacks, or impersonate legitimate devices. This is a significant security concern.
- Encryption: ChaCha20-Poly1305 AEAD: ChaCha20-Poly1305 AEAD is a robust symmetric encryption algorithm that provides both confidentiality and integrity. It encrypts the data and includes an authentication tag, ensuring that the data remains confidential and has not been tampered with during transmission.
- Hashing: BLAKE2s is a cryptographic hash function that is designed for efficiency and security. It is suitable for use in applications where fast hashing is required. In this context, BLAKE2s might be used for hashing to verify the integrity of the data.
- Security Implications: The use of X25519 for key exchange provides a secure way for the communicating parties to agree on a shared secret, contributing to the confidentiality of the data. The lack of authentication is a significant security concern. Without authentication, the system is vulnerable to various attacks, including unauthorized access and man-in-the-middle attacks. The choice of ChaCha20-Poly1305

AEAD for encryption is positive, as it ensures both confidentiality and integrity of the transmitted data. BLAKE2s, if used for hashing, can contribute to the verification of data integrity.

NK Pattern

Analyzing the security properties of the NK pattern with the given specifications:

- Key Exchange: X25519 is a modern and secure elliptic curve Diffie-Hellman (ECDH) key exchange algorithm. It allows two parties to securely agree on a shared secret over an insecure communication channel. The use of X25519 for key exchange is a strong choice and contributes to the confidentiality of the data.
- Authentication: HMAC-SHA256 is a robust choice for authentication. HMAC (Hash-based Message Authentication Code) provides integrity and authenticity, and SHA256 is a secure hash function. This combination ensures that the data has not been tampered with and comes from a legitimate source.
- None (No Encryption): The absence of encryption means that the data is transmitted in plaintext. While this simplifies the communication process, it exposes the information to potential eavesdropping. Depending on the nature of the transmitted data, the lack of encryption may or may not be a concern.
- Hashing: SHA256 is a widely used and secure hash function. Its role in this context is likely for integrity verification. Hashing ensures that the data has not been altered during transmission.
- Security Implications: The use of X25519 for key exchange provides a strong foundation for secure communication, contributing to the confidentiality of the data. HMAC-SHA256 provides robust authentication, ensuring both integrity and authenticity of the exchanged messages. The lack of encryption means that the data is exposed during transmission. Depending on the use case, this may be acceptable or may pose a significant security risk. The choice of SHA256 for hashing aligns with contemporary security standards and is suitable for integrity verification.

IK / KX / XX Patterns Pattern

Analyzing the security properties of the IK / KX / XX pattern with the given specifications:

1. Main Security Features:

  - Key Exchange: X25519 is a secure elliptic curve Diffie-Hellman (ECDH) key exchange algorithm, providing a secure method for two parties to agree on a shared secret over an insecure channel. The use of X25519 for key exchange is a strong choice, contributing to the confidentiality of the data.
  - No Authentication: The absence of authentication means there is no mutual verification of the communicating parties. Without proper authentication, the system is vulnerable to various attacks, including unauthorized access and man-in-the-middle attacks. This is a significant security concern.
  - Encryption: ChaCha20-Poly1305 AEAD is a strong symmetric encryption algorithm that provides both confidentiality and integrity. It encrypts the data and includes an authentication tag, ensuring that the data remains confidential and has not been tampered with during transmission.
  - Hashing: BLAKE2s: BLAKE2s is a cryptographic hash function designed for efficiency and security. It is suitable for applications where fast hashing is required. In this context, BLAKE2s might be used for hashing to verify the integrity of the data.
  - Security Implications: The use of X25519 for key exchange contributes to the confidentiality of the data, as it enables the parties to establish a shared secret securely. The absence of authentication is a significant security concern. Lack of authentication means that there is no mutual verification, leaving the system vulnerable to various security threats. ChaCha20-Poly1305 AEAD provides strong encryption, ensuring both confidentiality and integrity of the transmitted data. The choice of BLAKE2s for hashing, if used for integrity verification, aligns with contemporary security standards.

2. The difference between these patterns:
   (a) IK (Initiator to Responder with Key Confirmation):
      - Initiator Sends: The initiator sends its public key and some optional data.
      - Responder Sends: The responder sends its public key, some optional data, and a payload that includes key confirmation.
      - Key Confirmation: Key confirmation is an essential aspect of the IK pattern, ensuring both parties have agreed on the same symmetric keys securely.

   (b) KX (Key Exchange):

      - Initiator Sends: The initiator sends its public key and some optional data.
      - Responder Sends: The responder sends its public key and some optional data.
      - Key Exchange: The key exchange pattern involves both parties contributing public keys, and the protocol computes shared secrets based on these public keys. It typically doesn't include built-in key confirmation.

   (c) XX (Key Exchange with Cross-Key Confirmation):

      - Both Parties Send: Both the initiator and responder send their public keys and some optional data.
      - Key Exchange: Like the KX pattern, the XX pattern involves both parties contributing public keys, and shared secrets are derived.
      - Cross-Key Confirmation: The XX pattern includes a cross-key confirmation mechanism, which means both parties confirm that they have derived the same shared secrets.

3. Overall Shared Security Properties:

   - All three patterns (IK, KX, XX) can use similar cryptographic algorithms for key exchange, authentication, encryption, and hashing, as specified in the NPF or other relevant specifications.
   - The specified security properties (e.g., X25519 for key exchange, ChaCha20-Poly1305 for encryption, BLAKE2s for hashing) can be used consistently across these patterns.

4. Overall Key Differences:

   - The primary differences lie in the communication flow and whether key confirmation is explicitly built into the pattern.
   - IK includes key confirmation as a distinct step, ensuring that both parties have successfully derived the same symmetric keys.
   - KX typically focuses on the exchange of public keys and derivation of shared secrets but may not include a built-in mechanism for key confirmation.

  - XX, like KX, involves key exchange but also includes a cross-key confirmation mechanism, ensuring agreement on shared secrets between both parties.

NX Pattern

Analyzing the XN pattern with the given security properties:

- Key Exchange: X25519 is a secure elliptic curve Diffie-Hellman (ECDH) key exchange algorithm. It allows two parties to establish a shared secret over an insecure communication channel. The use of X25519 for key exchange contributes to the confidentiality of the data.
- Authentication: HMAC-SHA256 is a robust choice for authentication. HMAC provides integrity and authenticity, and SHA256 is a secure hash function. This combination ensures that the data has not been tampered with and comes from a legitimate source.
- Encryption: ChaCha20-Poly1305 AEAD is a strong symmetric encryption algorithm that provides both confidentiality and integrity. It encrypts the data and includes an authentication tag, ensuring that the data remains confidential and has not been tampered with during transmission.
- Hashing: SHA256 is a widely used and secure hash function. Its role in this context is likely for integrity verification. Hashing ensures that the data has not been altered during transmission.
- Security Implications: The use of X25519 for key exchange contributes to the confidentiality of the data by allowing the parties to establish a shared secret securely. HMAC-SHA256 provides robust authentication, ensuring both integrity and authenticity of the exchanged messages. ChaCha20-Poly1305 AEAD offers strong encryption, ensuring both confidentiality and integrity of the transmitted data. The choice of SHA256 for hashing aligns with contemporary security standards and is suitable for integrity verification.

IX Pattern

Analyzing the IX pattern with the given security properties:

- Key Exchange: X25519 is a modern and secure elliptic curve Diffie-Hellman (ECDH) key exchange algorithm. It enables two parties to agree on a shared secret over an insecure communication channel. The use of X25519 for key exchange is a strong choice, contributing to the confidentiality of the data.

- No Authentication: The absence of authentication means there is no mutual verification of the communicating parties. Without authentication, the system is vulnerable to various attacks, including unauthorized access and man-in-the-middle attacks. This is a significant security concern.
- Encryption: ChaCha20-Poly1305 AEAD is a strong symmetric encryption algorithm that provides both confidentiality and integrity. It encrypts the data and includes an authentication tag, ensuring that the data remains confidential and has not been tampered with during transmission.
- Hashing: BLAKE2s is a cryptographic hash function designed for efficiency and security. It is suitable for applications where fast hashing is required. In this context, BLAKE2s might be used for hashing to verify the integrity of the data.
- Security Implications: The use of X25519 for key exchange provides a strong foundation for secure communication, contributing to the confidentiality of the data. The absence of authentication is a significant security concern. Lack of authentication means that there is no mutual verification, leaving the system vulnerable to various security threats. ChaCha20-Poly1305 AEAD provides strong encryption, ensuring both confidentiality and integrity of the transmitted data. The choice of BLAKE2s for hashing, if used for integrity verification, aligns with contemporary security standards.

XN / KN Pattern

Analyzing the XN and KN patterns with the given security properties:

1. Main Security Features:
    - Key Exchange: X25519 is a modern and secure elliptic curve Diffie-Hellman (ECDH) key exchange algorithm. It allows two parties to establish a shared secret over an insecure communication channel, contributing to the confidentiality of the data.
    - No Authentication: The absence of authentication means there is no mutual verification of the communicating parties. This lack of authentication poses a significant security concern, leaving the system vulnerable to various attacks, including unauthorized access and man-in-the-middle attacks.

  - Encryption: ChaCha20-Poly1305 AEAD is a strong symmetric encryption algorithm that provides both confidentiality and integrity. It encrypts the data and includes an authentication tag, ensuring that the data remains confidential and has not been tampered with during transmission.
  - Hashing: BLAKE2s is a cryptographic hash function designed for efficiency and security. It is suitable for applications where fast hashing is required. In this context, BLAKE2s might be used for hashing to verify the integrity of the data.
  - Security Implications: The use of X25519 for key exchange provides a strong foundation for secure communication, contributing to the confidentiality of the data. The absence of authentication is a significant security concern. Lack of authentication means that there is no mutual verification, leaving the system vulnerable to various security threats. ChaCha20-Poly1305 AEAD provides strong encryption, ensuring both confidentiality and integrity of the transmitted data. The choice of BLAKE2s for hashing, if used for integrity verification, aligns with contemporary security standards.

2. The differences between the XN and KN patterns:
   (a) XN Pattern:
      - Initiator Sends: The initiator sends its public key and some optional data.
      - Responder Sends: The responder sends its public key, some optional data, and a payload that includes key confirmation.
      - Key Confirmation: Key confirmation is an essential aspect of the XN pattern, ensuring both parties have agreed on the same symmetric keys securely.

   (b) KN Pattern:
      - Initiator Sends: The initiator sends its public key and some optional data.
      - Responder Sends: The responder sends its public key and some optional data.
      - Key Exchange: The key exchange pattern involves both parties contributing public keys, and the protocol computes shared secrets based on these public keys. It typically doesn't include built-in key confirmation.

(c) Differences:

- The key difference lies in the presence of key confirmation. XN includes a specific payload for key confirmation, ensuring both parties agree on the same symmetric keys securely. KN, on the other hand, typically focuses on key exchange without built-in confirmation.
- The XN pattern is designed to explicitly handle key confirmation as part of the communication flow, providing an extra layer of security assurance.

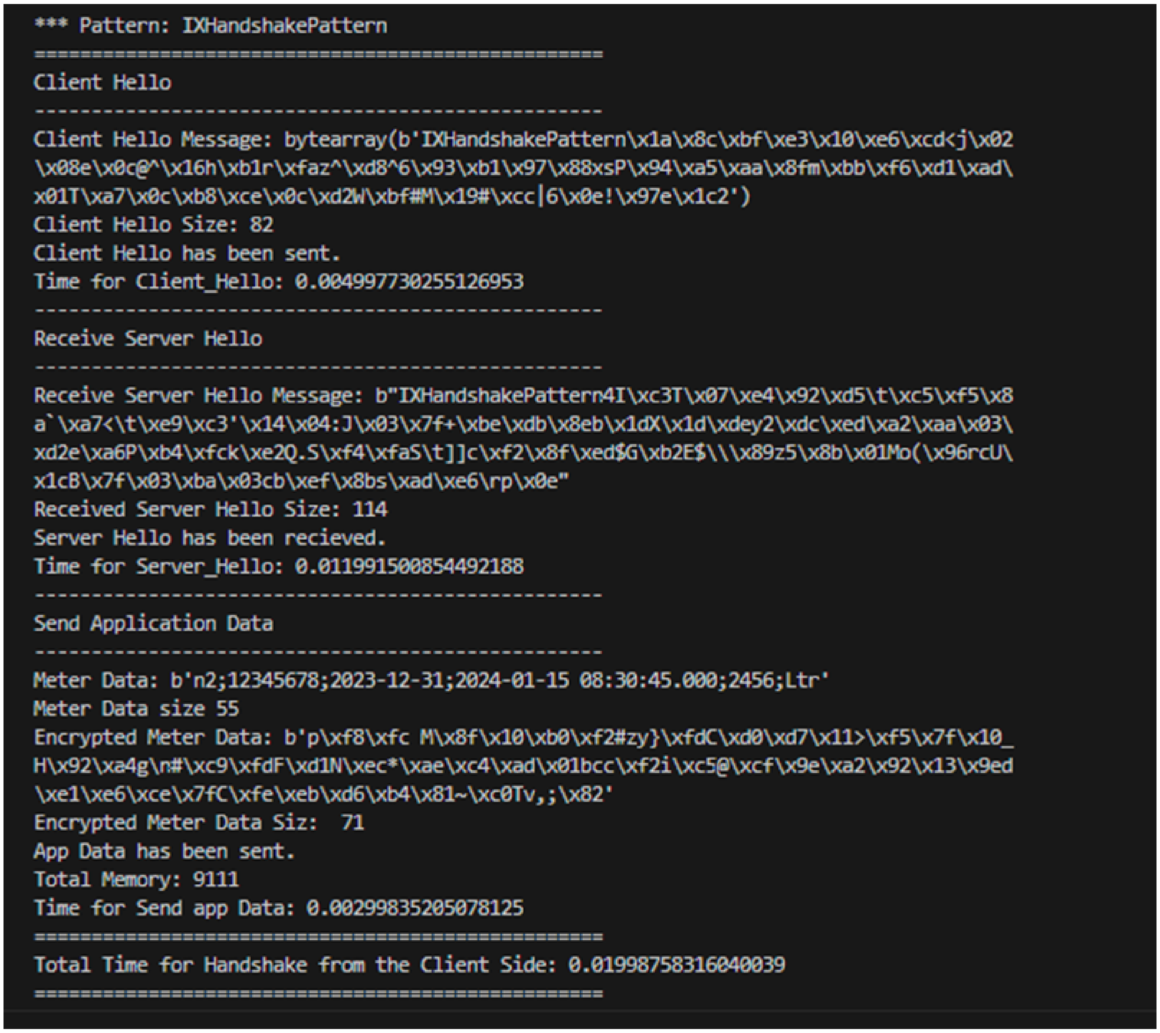
5.2.2 NPF Implementation

In the integration of the NPF within a Python-based Client/Server architecture, we meticulously employed six distinct handshake patterns—IK, KX, XX, NX, IX, and KN—to lay the groundwork for our empirical analysis. This analysis entailed a rigorous execution of each pattern to assess their inherent security attributes while closely monitoring memory consumption and packet dimensions throughout each phase of the handshake process. Our methodology differentiated between two primary message exchange paradigms: a three-message sequence and a two-message counterpart. For each paradigm, we meticulously recorded memory utilization and temporal metrics at pivotal junctures, including the ClientHello and ServerHello phases, across both client and server entities. Moreover, the analysis accounted for the total duration requisite for the bidirectional flow of application data, with a special emphasis on an additional step in the three-message sequence that precedes the server's data processing phase. The empirical findings, rooted in the systematic evaluation of the selected patterns (IK, KX, XX, NX, IX, KN), are concisely tabulated in Table 5.3. To facilitate a nuanced comparative analysis, we have also crafted detailed visual representations illustrating variations in memory usage, time elapsed, and packet sizes across the different patterns. Figure 5.2 provides a glimpse into the terminal interface during the execution of the IX pattern, offering a practical illustration of the protocol in action.

The memory usage for the six patterns as shown in the figure 5.3, elapsed time showing in 5.4, and the packet size showing in figure 5.5. As well showing the total memory usage and time for both server and client in figure5.6 and figure 5.7. The first three patterns (IX, KN, KX) show that both the client and server use an equal amount of memory. Starting from the NK pattern, there is a difference in memory usage between the client and server, with the server generally using less memory than the client, except for the NX pattern where both are equal. Peak Memory Usage: The XX pattern shows the highest memory usage for the client at 373, indicating that this pattern may be the most memory-intensive for the client.

Table 5.2: Noise Framework Predefined Patterns – Security Conclusion

No	Pattern	Security Conclusion
1	XN	The XN pattern lack of authentication and encryption is crucial for securing the wireless communication between a meter and a gateway via Wireless M-Bus. Implementing robust security measures will help protect the integrity, confidentiality, and authenticity of the transmitted data, safeguarding both the utility provider and the end-users.
2	KK	The KK pattern with ChaCha20-Poly1305 AEAD provides strong encryption, the lack of authentication and hashing introduces vulnerabilities that need to be addressed. Implementing proper authentication and integrity verification measures, along with careful key management, is essential for a more comprehensive and secure communication system.
3	NN	The NN pattern with key derivation from a pre-shared key, HMAC-SHA257 for authentication, no encryption, and SHA256 for hashing provides strong integrity and authenticity but lacks confidentiality. The security of the system depends heavily on the protection of the pre-shared key and the implementation of HMAC-SHA257.
4	IN	The IN pattern with X25519 for key exchange, HMAC-SHA256 for authentication, no encryption, and SHA256 for hashing provides a strong foundation for secure communication, with emphasis on integrity and authenticity. The security considerations should be tailored to the specific requirements and sensitivity of the data being exchanged.
5	XK	The XK pattern with X25519 for key exchange, ChaCha20-Poly1305 AEAD for encryption, and BLAKE2s for hashing provides some strong security features, the lack of authentication poses a significant risk. It is crucial to address this vulnerability by implementing authentication mechanisms to ensure the overall security of the communication system.
6	NK	The NK pattern with X25519 for key exchange, HMAC-SHA256 for authentication, no encryption, and SHA256 for hashing provides a strong foundation for integrity and authenticity. However, the absence of encryption exposes the data during transmission, so it's important to assess whether this aligns with the specific security needs of the application.
7	IK / KX / XX	The IK/KX/XX patterns with X25519 for key exchange, ChaCha20-Poly1305 AEAD for encryption, and BLAKE2s for hashing provide some strong security features. However, the lack of authentication poses a significant risk, and it's crucial to address this vulnerability to ensure the overall security of the communication system. The primary differences between IK, KX, and XX lie in the communication flow and the presence of key confirmation mechanisms. IK explicitly includes key confirmation, KX focuses on key exchange without built-in confirmation, and XX includes a cross-key confirmation mechanism. The specified security properties can be shared across these patterns, depending on the specific protocol design and requirements.
8	NX	The XN pattern with X25519 for key exchange, HMAC-SHA256 for authentication, ChaCha20-Poly1305 AEAD for encryption, and SHA256 for hashing provides a strong foundation for ensuring confidentiality, integrity, and authenticity of the exchanged data. However, as with any cryptographic system, it's crucial to stay vigilant about updates and best practices to maintain security over time.
9	IX	the IX pattern with X25519 for key exchange, ChaCha20-Poly1305 AEAD for encryption, and BLAKE2s for hashing provides some strong security features. However, the lack of authentication poses a significant risk, and it's crucial to address this vulnerability to ensure the overall security of the communication system.
10	KN	The KN pattern provides key exchange capabilities with ChaCha20-Poly1305 AEAD for encryption, BLAKE2s for hashing, and lacks built-in authentication. The specific security considerations and potential need for additional authentication mechanisms depend on the context and requirements of the cryptographic protocol.



```
*** Pattern: IXHandshakePattern
==================================================
Client Hello
--------------------------------------------------
Client Hello Message: bytearray(b'IXHandshakePattern\x1a\x8c\xbf\xe3\x10\xe6\xcd<j\x02
\x08e\x0c@^\x16h\xb1r\xfaz^\xd8^6\x93\xb1\x97\x88xsP\x94\xa5\xaa\x8fm\xbb\xf6\xd1\xad\
x01T\xa7\x0c\xb8\xce\x0c\xd2W\xbf#M\x19#\xcc|6\x0e!\x97e\x1c2')
Client Hello Size: 82
Client Hello has been sent.
Time for Client_Hello: 0.004997730255126953
--------------------------------------------------
Receive Server Hello
--------------------------------------------------
Receive Server Hello Message: b"IXHandshakePattern4I\xc3T\x07\xe4\x92\xd5\t\xc5\xf5\x8
a`\xa7<\t\xe9\xc3'\x14\x04:J\x03\x7f+\xbe\xdb\x8eb\x1dX\x1d\xdey2\xdc\xed\xa2\xaa\x03\
xd2e\xa6P\xb4\xfck\xe2Q.S\xf4\xfaS\t]]c\xf2\x8f\xed$G\xb2E$\\\x89z5\x8b\x01Mo(\x96rcU\
x1cB\x7f\x03\xba\x03cb\xef\x8bs\xad\xe6\rp\x0e"
Received Server Hello Size: 114
Server Hello has been recieved.
Time for Server_Hello: 0.011991500854492188
--------------------------------------------------
Send Application Data
--------------------------------------------------
Meter Data: b'n2;12345678;2023-12-31;2024-01-15 08:30:45.000;2456;Ltr'
Meter Data size 55
Encrypted Meter Data: b'p\xf8\xfc M\x8f\x10\xb0\xf2#zy}\xfdC\xd0\xd7\x11>\xf5\x7f\x10_
H\x92\xa4g\n#\xc9\xfdF\xd1N\xec*\xae\xc4\xad\x01bcc\xf2i\xc5@\xcf\x9e\xa2\x92\x13\x9ed
\xe1\xe6\xce\x7fC\xfe\xeb\xd6\xb4\x81~\xc0Tv,;\x82'
Encrypted Meter Data Siz:  71
App Data has been sent.
Total Memory: 9111
Time for Send app Data: 0.00299835205078125
==================================================
Total Time for Handshake from the Client Side: 0.01998758316040039
==================================================
```

Figure 5.2: Sample of a terminal view of the NPF communication for IX Pattern

Minimum Memory Usage: The KN pattern shows the lowest memory usage for both the client and server, at 187, suggesting that it is the least memory-intensive pattern among those presented. The XX pattern may be less suited for memory-constrained environments, especially on the client side. As for the elapse time, the first three patterns (IX, KN, and KX), the client and server have identical elapsed times for the execution of the patterns. Starting with the NK pattern, there is a difference in the elapsed times for the client and server, with the server generally completing the pattern in less time than the client. The XX pattern has the highest elapsed time for the client at 373 seconds, which is considerably higher than the server's time for the same pattern at 246 seconds. This suggests that the XX pattern is the most time-intensive, particularly for the client. The KN pattern is completed in the least amount of time by both the client and the server, each taking 187 seconds. This

Table 5.3: Noise Framework presenting the six patterns with the result as in memory usage, time, and packet size

No	Pattern	Terminal	Memory Usage	Total Time	Total Packet Size	Time ClientHello	Packet Size Byte	Time ServerHello	Packet Size Byte	Time ClientFinish	Packet Size ClientFinish	Time App Data	Meter DataSize	Packet Size Byte
1	**XX**	Server	11818	0.020986	246	0.002995	50	0.003998	114	0.005999	82			
	3 Msg	Client	11560	0.016989	373	0.002997	94	0.006993	114	0.002998	94	0.001999	55	71
		Total	**23378**	**0.03797**	**619**									
2	**IX**	Server	9003	0.019988	267	0.007996	82	0.005997	114			0.005997		71
	2 Msg	Client	9111	0.019988	267	0.004998	82	0.011992	114			0.002998	55	71
		Total	**18114**	**0.03998**	**534**									
3	**KN**	Server	8253	0.015992	187	0.006995	50	0.003998	66			0.004999		71
	2Msg	Client	8105	0.013992	187	0.002998	50	0.008993	66			0.002	55	71
		Total	**16358**	**0.02998**	**374**									
4	**KX**	Server	7506	0.017989	235	0.005998	50	0.005995	114			0.005997		71
	2 Msg	Client	7463	0.014982	235	0.001998	50	0.010993	114			0.001992	55	71
		Total	**14969**	**0.03297**	**470**									
5	**NK**	Server	6930	0.014989	203	0.007993	66	0.003997	66			0.002999		71
	2Msg	Client	6919	0.013992	203	0.003998	66	0.007994	66			0.002	55	71
		Total	**13849**	**0.02898**	**406**									
6	**NX**	Server	6641	0.017989	235	0.008996	50	0.003998	114			0.004996		71
	2 Msg	Client	6838	0.015989	235	0.003	50	0.010993	114			0.001996	55	71
		Total	**13479**	**0.03398**	**470**									

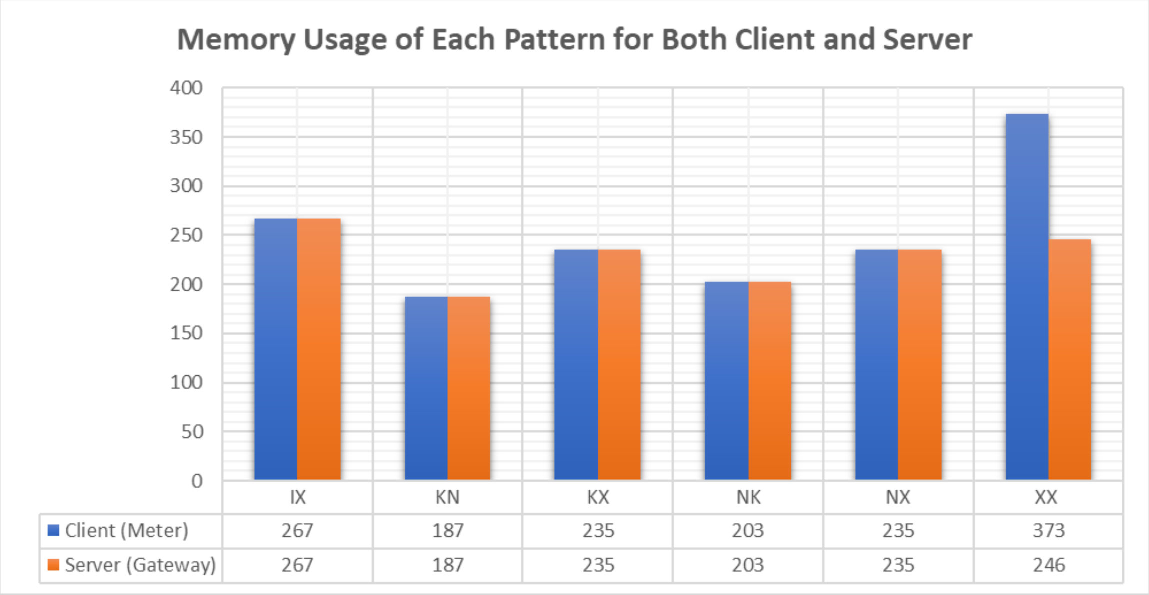
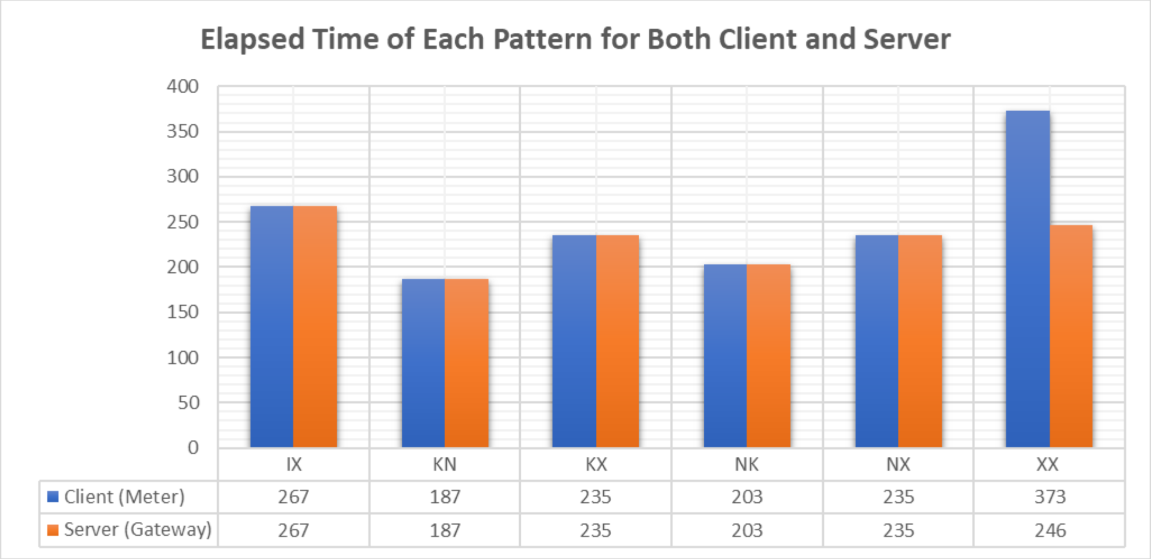


Figure 5.3: Total Memory Usage of the Six Patterns for both Client and Server



	IX	KN	KX	NK	NX	XX
■ Client (Meter)	267	187	235	203	235	373
■ Server (Gateway)	267	187	235	203	235	246

Figure 5.4: Total Elapsed Time of the Six Patterns for both Client and Server

indicates that KN may be the most efficient pattern in terms of time taken for execution. There is no clear increasing or decreasing trend across the patterns, but there is a notable spike in time for the client side on the XX pattern. This suggests that the XX pattern might be more complex or computationally intensive for the client compared to other patterns.

For the entire handshake process the memory usage across all patterns is relatively close in range, with the smallest memory usage for a pattern being just under 0.03 and the largest just under 0.04 of the unit used.

- Pattern IX: This pattern has the highest memory usage, at approximately 0.03997.
- Pattern KN: This shows the lowest memory usage, at approximately 0.02998, it is the most memory-efficient pattern among those presented.
- Pattern KX: It has a memory usage close to that of KN, at approximately 0.03297.
- Patterns NK and NX: Both these patterns have nearly identical memory usage, around 0.03397, indicating similar memory demands.
- Pattern XX: This has a slightly higher memory usage than NK and NX but is less than IX, at approximately 0.03797.

There is no clear ascending or descending trend in memory usage across the patterns. However, IX and XX patterns stand out as having the highest memory requirements, while KN

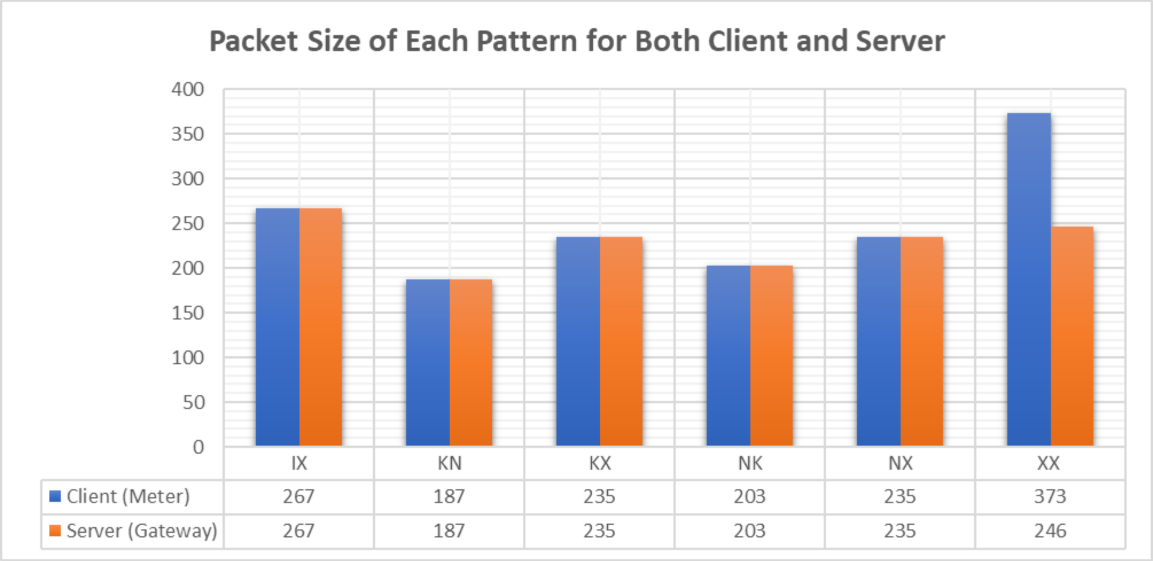


Figure 5.5: Enter Caption

appears to be the most memory efficient. f memory usage is a critical factor in system design, the KN pattern may be preferable. Conversely, if IX or XX patterns provide essential security features that outweigh memory considerations, they might still be chosen despite higher memory usage.

5.3 Phase Two: wM-Bus Protocol

In this phase, the primary objective was to establish a foundational performance baseline for the wM-Bus Protocol, adhering to the protocol standard. This assessment encompassed both hardware and software components.

5.3.1 Hardware Configuration

The initial step involved configuring the hardware, utilizing two wireless communication cards to emulate the roles of the Gateway (Server) and Meter (Client). For this experiment, the Radiocrafts developer kit, specifically the RC17xx line – MBUS4, was employed. These cards, part of the RC17xx line, namely MBUS4, support N-mode 169 MHz and are available in both High Power (27dBm) and Very High Power (30dBm) versions, as detailed in Table 5.4.

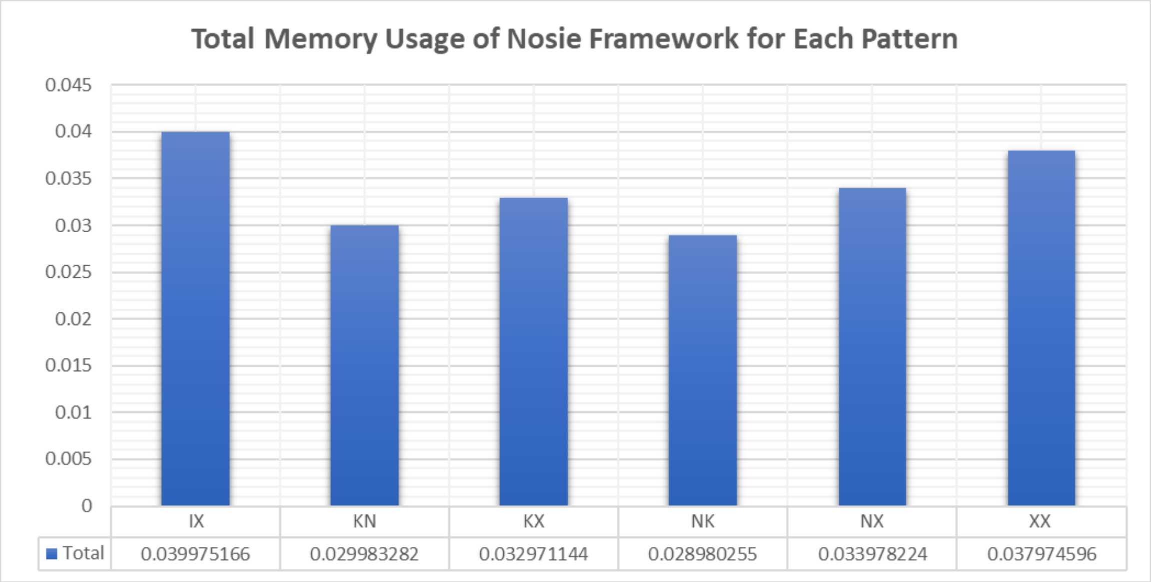


Figure 5.6: Total Memory Usage of each NPF Pattern

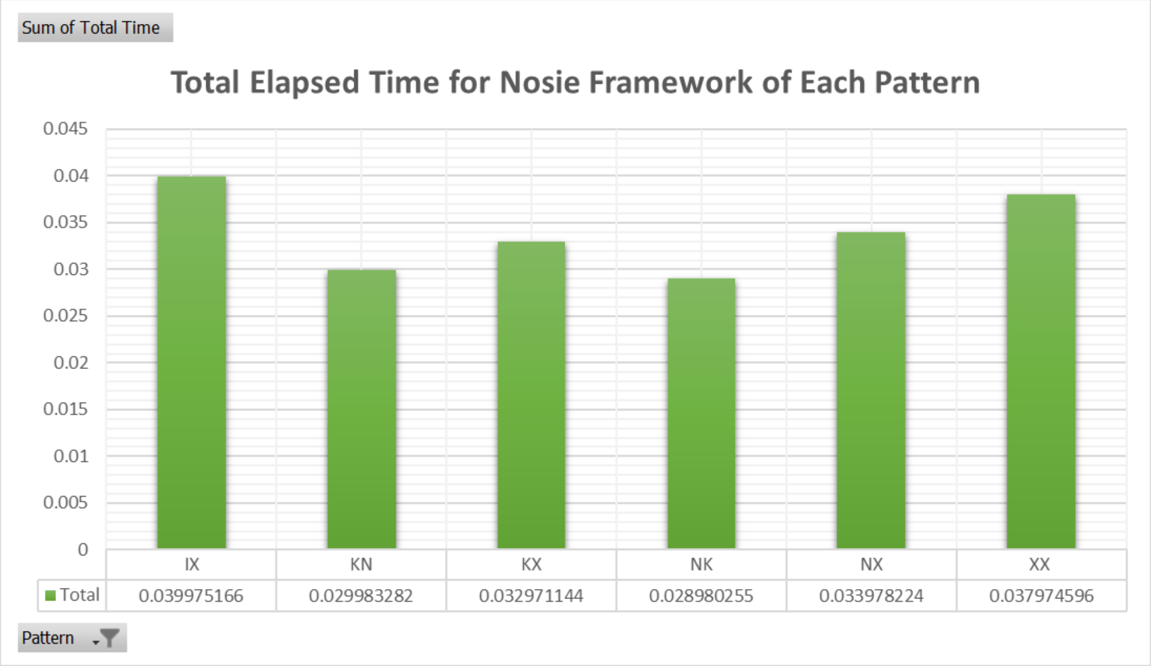
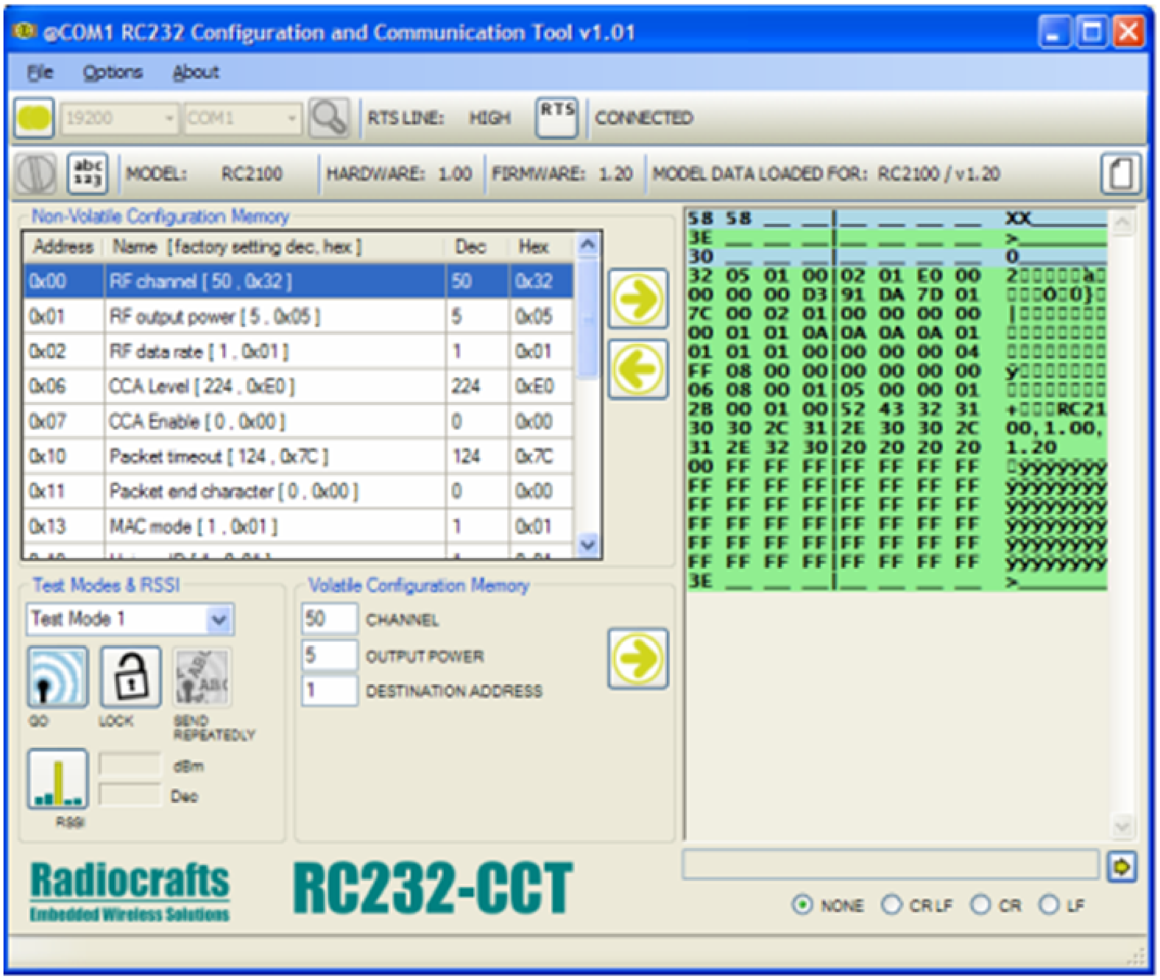


Figure 5.7: Total Elapsed Time of each NPF Pattern

Table 5.4: Wireless M-Bus Radiocrafts DK RC1701HP-MBUS4 Specifications

Module	Frequency band [MHz]	Radio channels [#]	Data rate [kbps]	RX sensitivity [dBm]	RX current [mA]	Pout [dBm]	TX current [mA]	SLEEP current [uA]	Operating supply voltage [V]	Operating temperature [deg C]	Indicative LOS [m]	Narrow Band
RC1701HP-MBUS4	169	10	2.4/ 4.8/ 19.2	-119/ -115/ -107	31	27	403	0,6	2,8 – 3,6	-30 to +85	20,000	Yes


@COM1 RC232 Configuration and Communication Tool v1.01
File
Options
About
RTS
Non-Volatile Configuration Memory
Test Modes & RSSI
GO
LOCK
SEND REPEATEDLY
RSSI
dBm
Dec
Volatile Configuration Memory
Radiocrafts
Embedded Wireless Solutions
RC232-CCT

19200 COM1 RTS LINE: HIGH CONNECTED

MODEL: RC2100 | HARDWARE: 1.00 | FIRMWARE: 1.20 | MODEL DATA LOADED FOR: RC2100 / v1.20

Address	Name [factory setting dec, hex]	Dec	Hex
0x00	RF channel [50 , 0x32]	50	0x32
0x01	RF output power [5 , 0x05]	5	0x05
0x02	RF data rate [1 , 0x01]	1	0x01
0x06	CCA Level [224 , 0xE0]	224	0xE0
0x07	CCA Enable [0 , 0x00]	0	0x00
0x10	Packet timeout [124 , 0x7C]	124	0x7C
0x11	Packet end character [0 , 0x00]	0	0x00
0x13	MAC mode [1 , 0x01]	1	0x01

Test Mode 1

50 CHANNEL
5 OUTPUT POWER
1 DESTINATION ADDRESS

```
58 58 __ __|__ __ __ __  XX______
3E __ __ __|__ __ __ __  >_______
30 __ __ __|__ __ __ __  0_______
32 05 01 00|02 01 E0 00  2□□□□□à□
00 00 00 D3|91 DA 7D 01  □□□O□0}□
7C 00 02 01|00 00 00 00  |□□□□□□□
00 01 01 0A|0A 0A 0A 01  □□□□□□□□
01 01 01 00|00 00 00 04  □□□□□□□□
FF 08 00 00|00 00 00 00  ÿ□□□□□□□
06 08 00 01|05 00 00 01  □□□□□□□□
2B 00 01 00|52 43 32 31  +□□□RC21
30 30 2C 31|2E 30 30 2C  00,1.00,
31 2E 32 30|20 20 20 20  1.20
00 FF FF FF|FF FF FF FF  □ÿÿÿÿÿÿÿ
FF FF FF FF|FF FF FF FF  ÿÿÿÿÿÿÿÿ
FF FF FF FF|FF FF FF FF  ÿÿÿÿÿÿÿÿ
FF FF FF FF|FF FF FF FF  ÿÿÿÿÿÿÿÿ
FF FF FF FF|FF FF FF FF  ÿÿÿÿÿÿÿÿ
FF FF FF FF|FF FF FF FF  ÿÿÿÿÿÿÿÿ
3E __ __ __|__ __ __ __  >_______
```

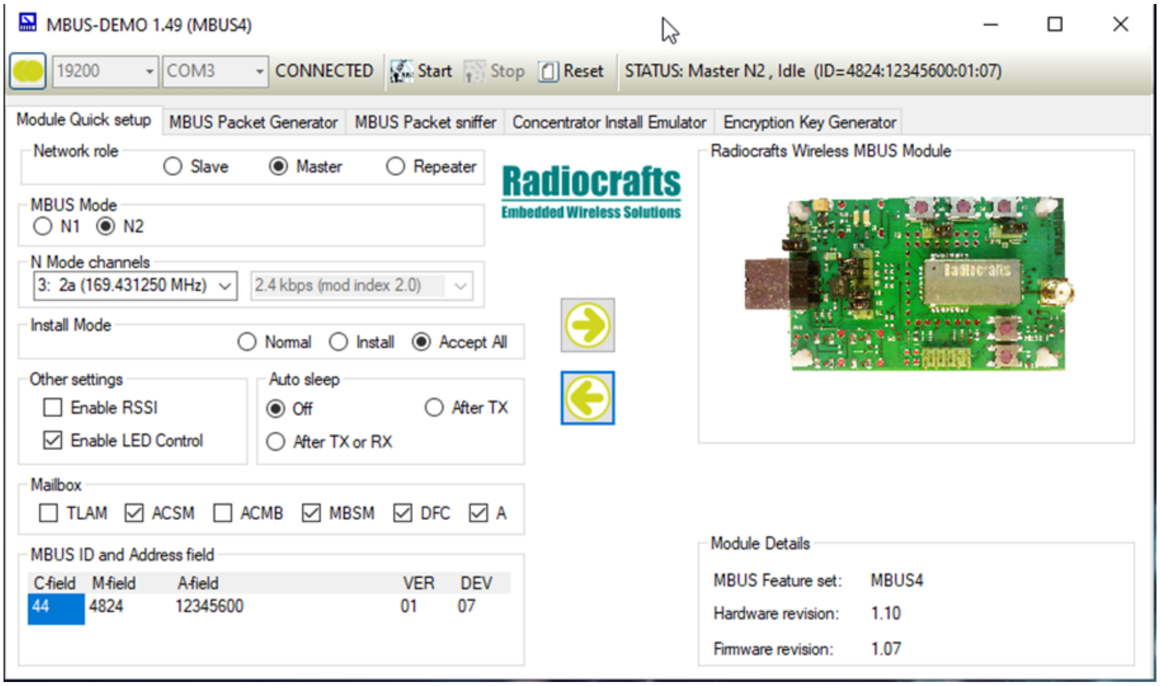
● NONE ○ CR LF ○ CR ○ LF

Figure 5.8: wM-Bus Development Kit – Software Tool RC232-CCT for Configuration

5.3.2 Software Configuration

RCTools

The developer kit proved instrumental in facilitating the configuration and testing of the initial communication between the two wM-Bus Cards. This process was streamlined with the utilization of the RCTools application, an essential tool within the PC suite designed to support testing, development, and deployment of Radiocrafts modules. The RC Tools-MBUS encompasses PC tools tailored for the Wireless MBUS family, incorporating both the CCT tool and the Demo tool, as illustrated in Figure 5.8, 5.9, and 5.10. The seamless installation of RCTools, inclusive of .NET and USB Driver components, is accomplished by executing the provided setup file.



19200 COM3 CONNECTED

STATUS: Master N2 , Idle (ID=4824:12345600:01:07)

Network role: ○ Slave ● Master ○ Repeater

MBUS Mode: ○ N1 ● N2

N Mode channels: 3: 2a (169.431250 MHz) 2.4 kbps (mod index 2.0)

Install Mode: ○ Normal ○ Install ● Accept All

Other settings: ☐ Enable RSSI ☑ Enable LED Control

Auto sleep: ● Off ○ After TX ○ After TX or RX

Mailbox: ☐ TLAM ☑ ACSM ☐ ACMB ☑ MBSM ☑ DFC ☑ A

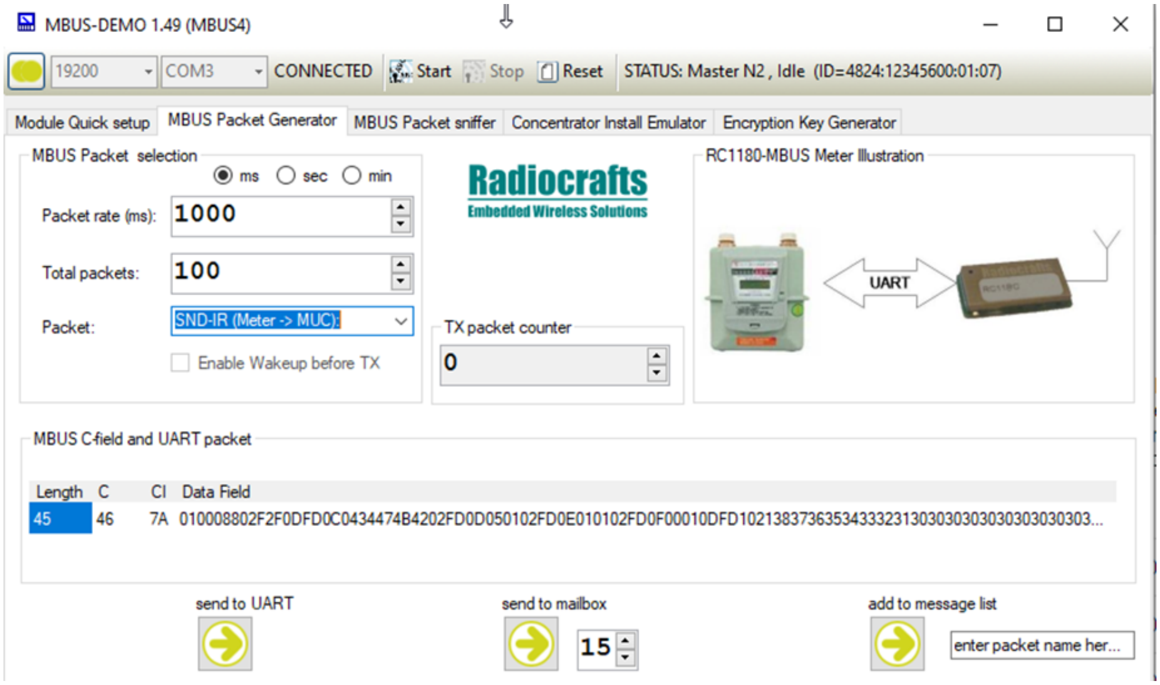
MBUS ID and Address field

C-field	M-field	A-field	VER	DEV
44	4824	12345600	01	07

Module Details

MBUS Feature set:	MBUS4
Hardware revision:	1.10
Firmware revision:	1.07

Figure 5.9: wM-Bus Development Kit – Software M-Bus-Demo for Testing



19200 COM3 CONNECTED

STATUS: Master N2 , Idle (ID=4824:12345600:01:07)

MBUS Packet selection: ● ms ○ sec ○ min

Packet rate (ms): 1000

Total packets: 100

Packet: SND-IR (Meter -> MUC)

☐ Enable Wakeup before TX

TX packet counter: 0

MBUS C-field and UART packet

Length	C	CI	Data Field
45	46	7A	010008802F2F0DFD0C0434474B4202FD0D050102FD0E010102FD0F00010DFD102138373635343332313030303030303030303...

send to mailbox: 15

add to message list: enter packet name her...

Figure 5.10: wM-Bus Development Kit – Software M-Bus-Demo for Testing

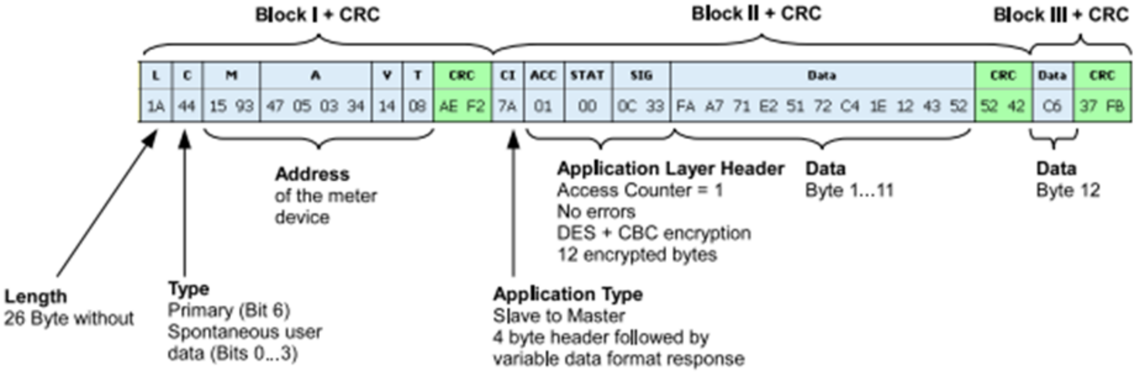


Figure 5.11: Example of wM-Bus Packet

5.3.3 wM-Bus Development Implementation

Implemented in Python, the development of the wM-Bus communication protocol provides insights into the interaction between two cards, simulating the roles of a meter(Client) and a gateway(server) within the system. Utilizing specific wM-Bus libraries, as well other libraries such as “tracemalloc” to calculate memory usage, as well “PySerial” to be able to connect the two cards and utilized to write and read from the serial devices. The software configuration for the development which set “COM3” and “COM4” for each card. The Bud-Rate for the communication is set at 12900. This implementation streamlines the process of establishing a connection and transmitting data. It’s crucial to emphasize that the payload transmission occurs in plaintext, devoid of any security measures. This intentional omission aims to create a baseline for wM-Bus communication on the gateway side without security layers, paving the way for future comparisons once security measures are integrated. The data transmission process follows the established wM-Bus fields detailed in 5.5, with the network packet frame visually depicted in Figure 5.11. The wM-Bus protocol encompasses various message types, including (CI), with specific examples such as 7A (Meter to Gateway), as outlined in Table (5). This comprehensive approach ensures a meticulous exploration of wM-Bus communication dynamics on the gateway and meter side, laying the foundation for subsequent assessments enhanced with security measures.

During the experiment, we systematically tested three distinct message types, namely ”7A,” ”72,” and ”08,” each exhibiting varied message lengths. A representative snippet of the code employed for this experimentation is illustrated in figure 5.12. During each iteration, the system dynamically generates a message with the designated length. As figure 5.13

Table 5.5: List of Wireless M-Bus fields

Name	Size (Byte)	Description
Length (L)	1	Number of packet bytes excluding the length field and all CRCs
Type (C)	1	Packet type
Manufacturer Id (M)	2	ID of the manufacturer code
Type (A)	6	Address of the meter device
Checksum (CRC)	2	Checksum of the current block
Application Type (CI)	1	Type field of the application layer
Application Layer	x	Data of the application layer

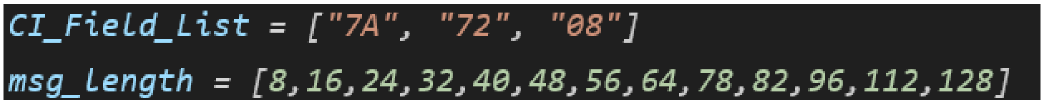
Table 5.6: List of Wireless M-Bus Message Types

Direction	Message Type	C-field	CA
Meter-to-Gateway	SND-IR (Meter ->MUC)	46	7A
Meter-to-Gateway	SND-NR (Meter ->MUC)	44	7A
Meter-to-Gateway	RSP-UD (Meter ->MUC)	8	72
Meter-to-Gateway	AES 1 block, short header	44	7A
Meter-to-Gateway	AES 1 block, long header	44	72
Meter-to-Gateway	Short Test Message	44	8
Meter-to-Gateway	Medium test message:	44	8
Meter-to-Gateway	Long test Message:	44	8

provides a glimpse of a sample meter log, showcasing the calculated memory usage and elapsed time associated with the process.

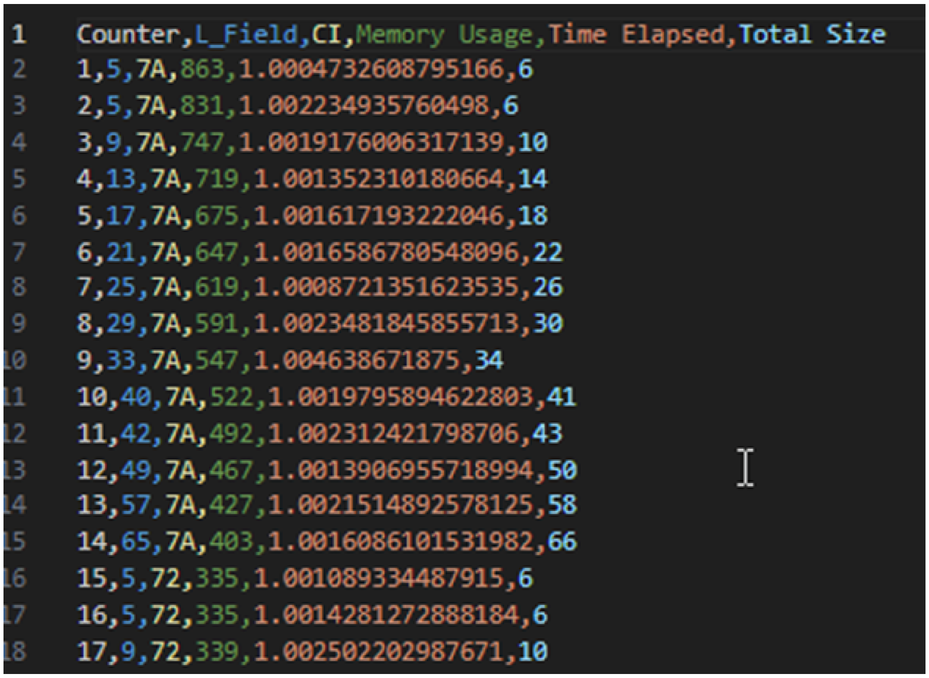
wM-Bus Gateway Implementation

The experimental results at the gateway side are depicted in three charts, each illustrating the memory usage for different message types, as seen in Figure 5.14. Notably, the initial point for each message type exhibits elevated memory usage readings. However, upon the initialization of all program variables, the memory usage experiences a decline and stabilizes after the message length surpasses 27 bytes. A noteworthy observation from the graph is that while message types 72 and 08 demonstrate a slight increase in memory usage as the length expands, message type 7A continues to exhibit a decreasing trend. Furthermore, in Figures (5.15, 5.16), the depicted time required to establish communication and transmit messages from the meter to the gateway reveals a remarkably close correlation,



```
CI_Field_List = ["7A", "72", "08"]
msg_length = [8,16,24,32,40,48,56,64,78,82,96,112,128]
```

Figure 5.12: Sample of python code using different type of message (CI) with different lengths



```
Counter,L_Field,CI,Memory Usage,Time Elapsed,Total Size
1,5,7A,863,1.0004732608795166,6
2,5,7A,831,1.002234935760498,6
3,9,7A,747,1.0019176006317139,10
4,13,7A,719,1.001352310180664,14
5,17,7A,675,1.001617193222046,18
6,21,7A,647,1.0016586780548096,22
7,25,7A,619,1.0008721351623535,26
8,29,7A,591,1.0023481845855713,30
9,33,7A,547,1.004638671875,34
10,40,7A,522,1.0019795894622803,41
11,42,7A,492,1.002312421798706,43
12,49,7A,467,1.0013906955718994,50
13,57,7A,427,1.0021514892578125,58
14,65,7A,403,1.0016086101531982,66
15,5,72,335,1.001089334487915,6
16,5,72,335,1.0014281272888184,6
17,9,72,339,1.002502202987671,10
```

Figure 5.13: Sample of the Meter(Client) log for the calculated memory usage and elapsed time

with the exception of message types 08 and 72. Notably, for a message length of 15 bytes, these specific message types exhibit considerably higher time requirements, introducing an interesting variation in the overall communication dynamics. To provide a more accurate depiction, Figure 5.2 now offers a glimpse of the terminal view specifically showcasing the last message length with message type ’08’. This display includes the raw message, alongside calculated metrics such as memory usage and time. Additionally, header fields such as version, manufacture ID, and payload length are visible, offering a more detailed insight into the gateway-side communication.

wM-Bus Meter Implementation

Regarding the meter side, the illustrated results are presented below in Figure (5.18, 5.19). These figures distinctly showcase the memory usage variations for different message types within the wM-Bus protocol at various lengths. It’s evident that the ’7A’ message type demands a higher memory consumption compared to the other two message types, namely 72 and 08. Noteworthy is the observation that both message types 08 and 72 exhibit a marginal increase in memory usage as the message size expands. Concerning the time required for the meter to transmit data across various message types and lengths, the insights provided by the charts in Figure (5.20, 5.21 and 16b offer a clear indication of time

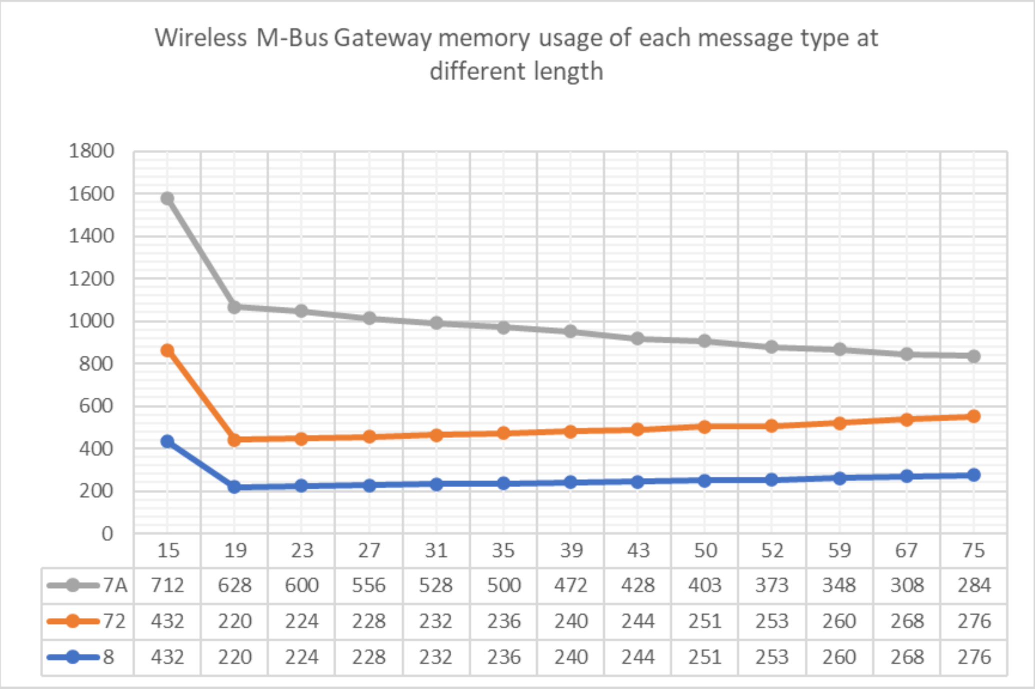


Figure 5.14: wM-Bus Memory Usage for different Message Types (7A, 72, 8) with different message length

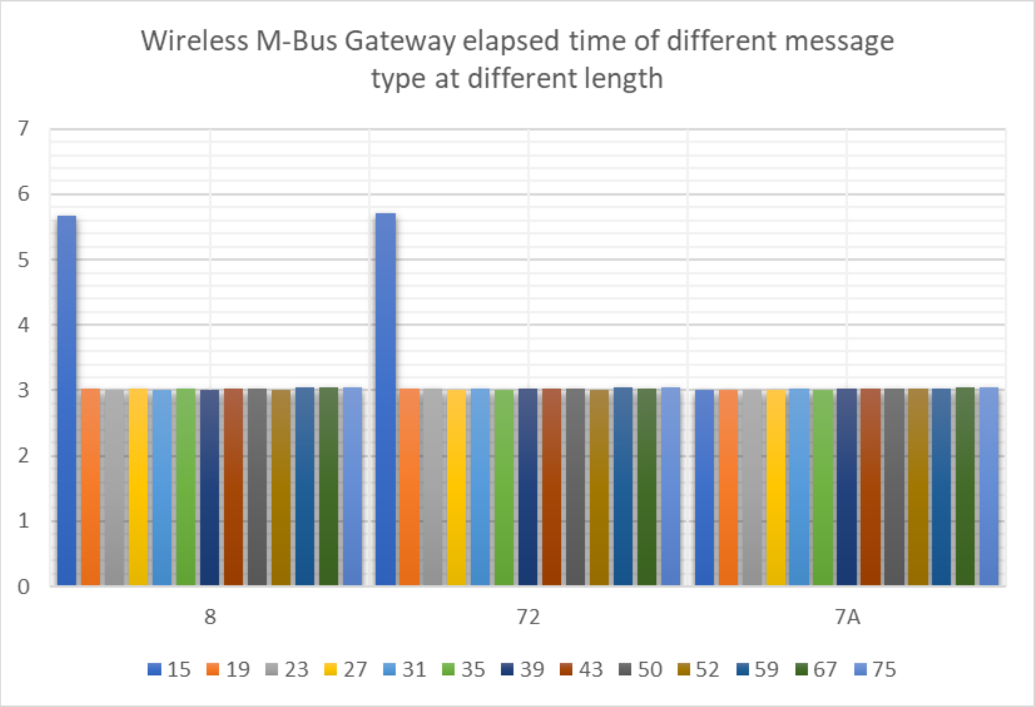


Figure 5.15: wM-Bus Time Elapsed for different Message Types (8, 72, 7A) with different message length

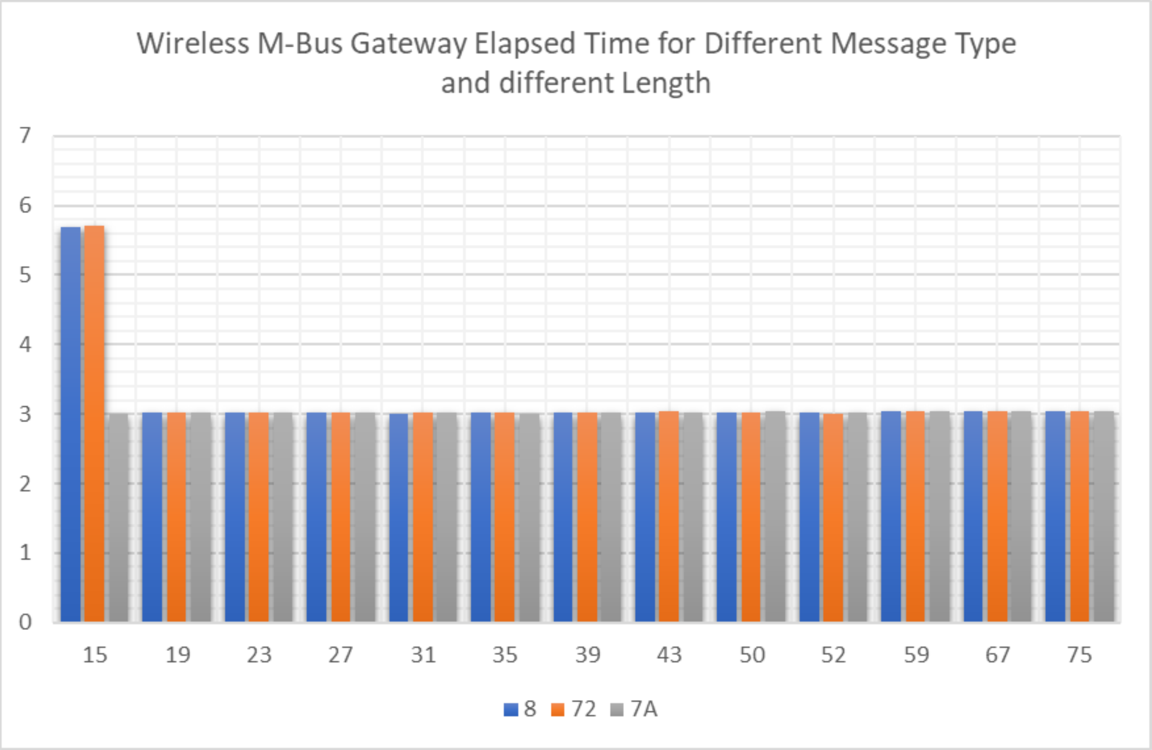
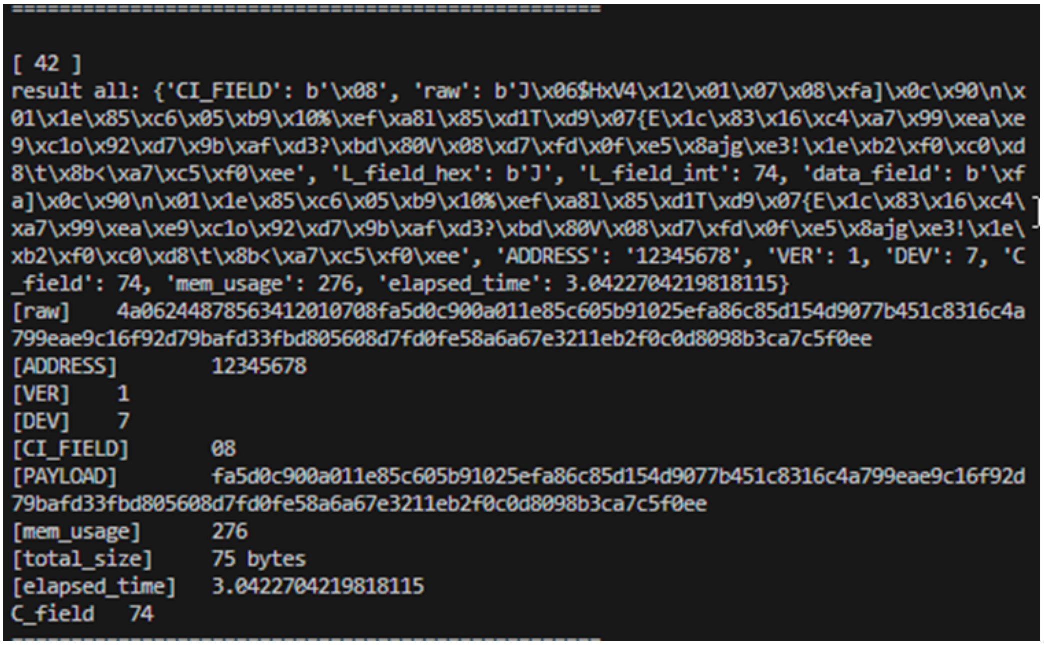


Figure 5.16: wM-Bus Time Elapsed for different Message Types (8, 72, 7A) with different message length



```
==================================================

[ 42 ]
result all: {'CI_FIELD': b'\x08', 'raw': b'J\x06$HxV4\x12\x01\x07\x08\xfa]\x0c\x90\n\x
01\x1e\x85\xc6\x05\xb9\x10%\xef\xa81\x85\xd1T\xd9\x07{E\x1c\x83\x16\xc4\xa7\x99\xea\xe
9\xc1o\x92\xd7\x9b\xaf\xd3?\xbd\x80V\x08\xd7\xfd\x0f\xe5\x8ajg\xe3!\x1e\xb2\xf0\xc0\xd
8\t\x8b<\xa7\xc5\xf0\xee', 'L_field_hex': b'J', 'L_field_int': 74, 'data_field': b'\xf
a]\x0c\x90\n\x01\x1e\x85\xc6\x05\xb9\x10%\xef\xa81\x85\xd1T\xd9\x07{E\x1c\x83\x16\xc4\
xa7\x99\xea\xe9\xc1o\x92\xd7\x9b\xaf\xd3?\xbd\x80V\x08\xd7\xfd\x0f\xe5\x8ajg\xe3!\x1e\
xb2\xf0\xc0\xd8\t\x8b<\xa7\xc5\xf0\xee', 'ADDRESS': '12345678', 'VER': 1, 'DEV': 7, 'C
_field': 74, 'mem_usage': 276, 'elapsed_time': 3.0422704219818115}
[raw]    4a06244878563412010708fa5d0c900a011e85c605b91025efa86c85d154d9077b451c8316c4a
799eae9c16f92d79bafd33fbd805608d7fd0fe58a6a67e3211eb2f0c0d8098b3ca7c5f0ee
[ADDRESS]        12345678
[VER]    1
[DEV]    7
[CI_FIELD]       08
[PAYLOAD]        fa5d0c900a011e85c605b91025efa86c85d154d9077b451c8316c4a799eae9c16f92d
79bafd33fbd805608d7fd0fe58a6a67e3211eb2f0c0d8098b3ca7c5f0ee
[mem_usage]      276
[total_size]     75 bytes
[elapsed_time]   3.0422704219818115
C_field   74
```

Figure 5.17: Terminal View – Example of the message “08” with length 75 Bytes. Showing the raw message and the packet header fields for the gateway

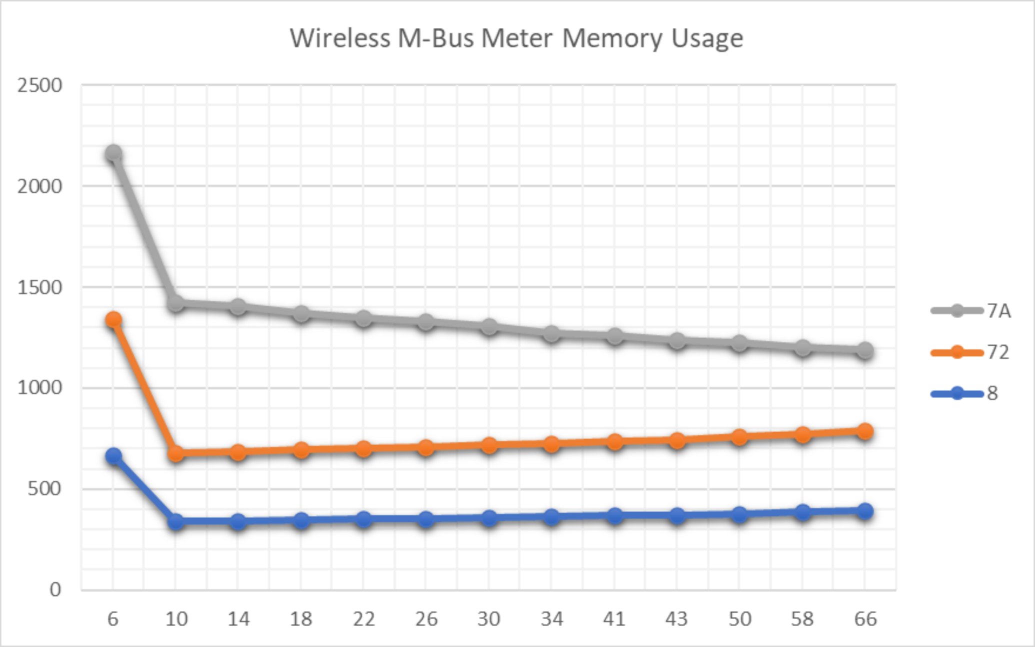


Figure 5.18: wM-Bus meter memory usage for three different messages at different length

stability. Remarkably, the time remains closely aligned across different message lengths, underscoring a consistent transmission pattern. Notably, in the initial run with a message size of 6 bytes, there is a conspicuous elevation in time, despite the reading being for the second run. However, subsequent iterations demonstrate a reduction in the time required for data transmission. In particular, message type 7A consistently showcases a reduction in the time needed for transmission throughout the runs. Meanwhile, for message types 08 and 72, there is a marginal increase in time, although not substantial. This nuanced analysis provides valuable insights into the temporal dynamics of data transmission from the meter side, revealing patterns of efficiency and responsiveness across diverse message types and lengths.

As shown in Figure 5.22 and Figure (5.23) respectively a comprehensive chart that combines both the gateway and meter aspects in terms of memory usage and elapsed time. It encapsulates the entire communication process, simulating the transmission of randomly generated messages from the meter to the gateway at varying lengths for each distinct message type. This complete view offers a consolidated perspective on the performance dynamics, considering both memory utilization and the time elapsed throughout the communication cycle between the meter and gateway.

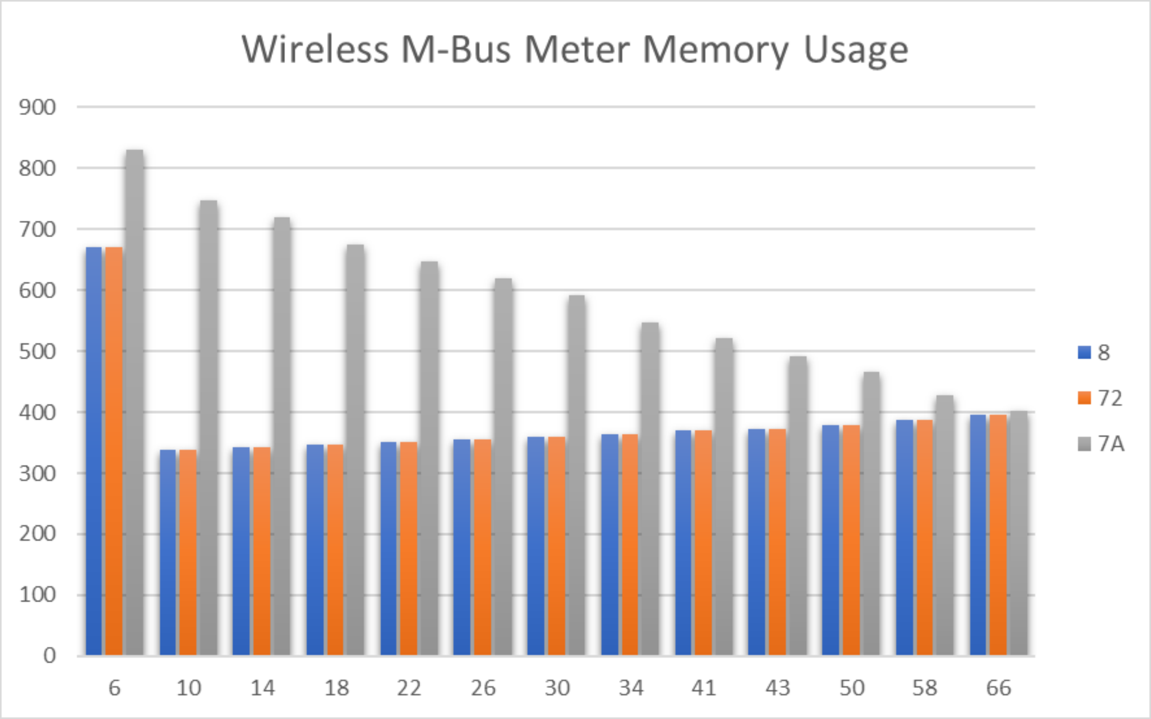
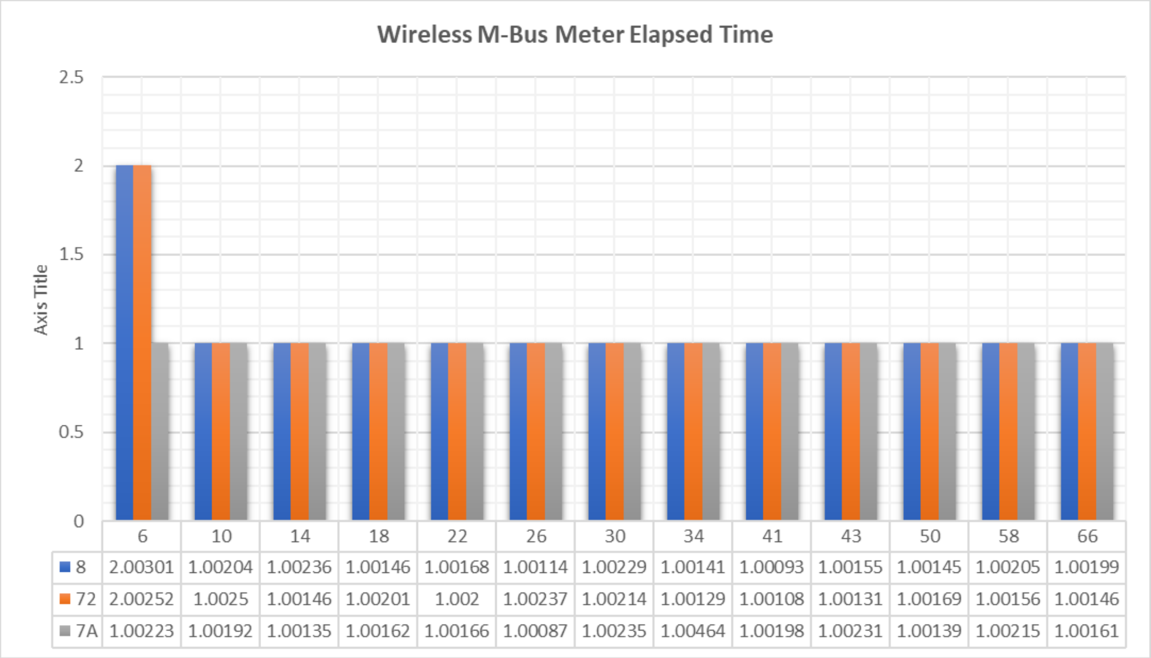


Figure 5.19: wM-Bus meter memory usage for three different messages at different length



	6	10	14	18	22	26	30	34	41	43	50	58	66
■ 8	2.00301	1.00204	1.00236	1.00146	1.00168	1.00114	1.00229	1.00141	1.00093	1.00155	1.00145	1.00205	1.00199
■ 72	2.00252	1.0025	1.00146	1.00201	1.002	1.00237	1.00214	1.00129	1.00108	1.00131	1.00169	1.00156	1.00146
■ 7A	1.00223	1.00192	1.00135	1.00162	1.00166	1.00087	1.00235	1.00464	1.00198	1.00231	1.00139	1.00215	1.00161

Figure 5.20: wM-Bus meter memory usage for three different messages at different length

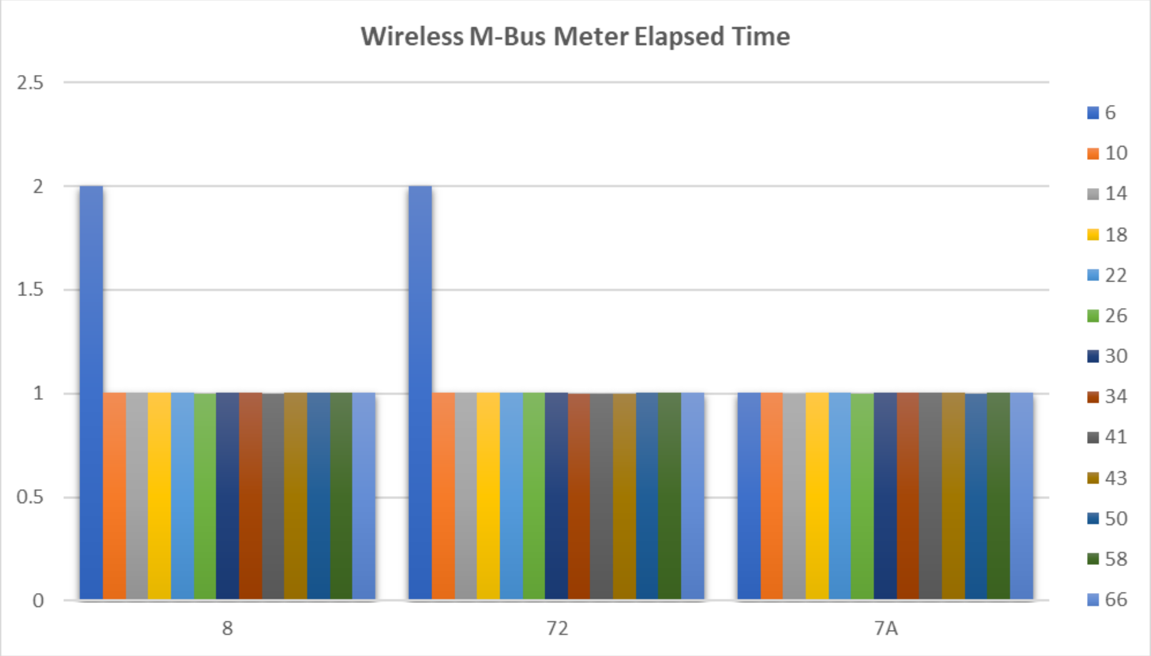


Figure 5.21: wM-Bus meter memory usage for three different messages at different length

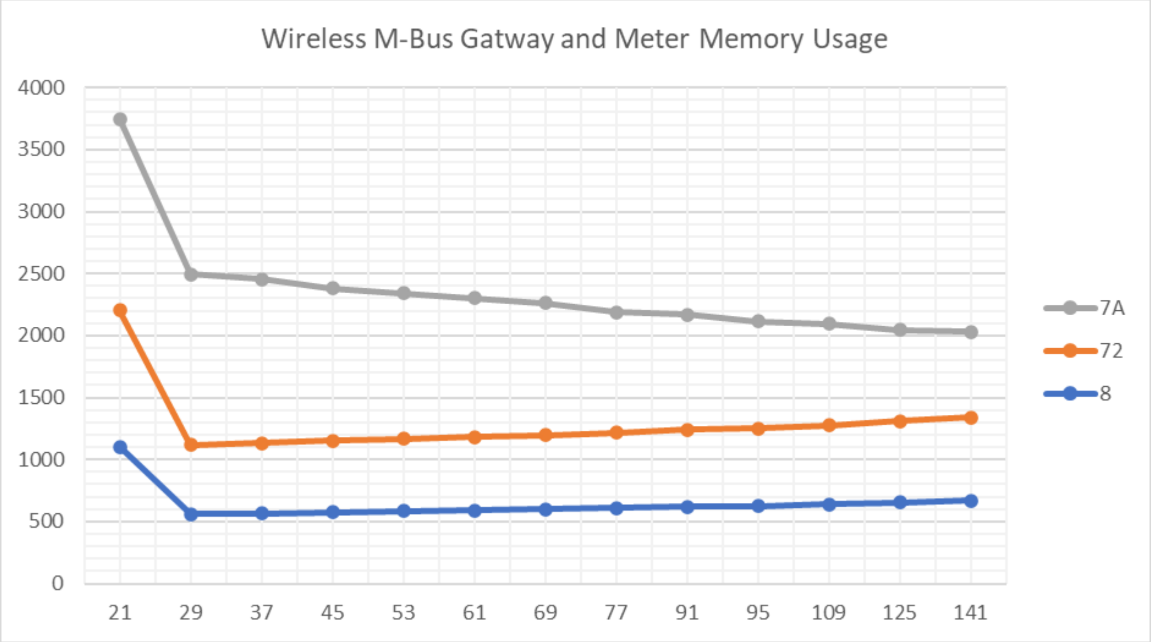


Figure 5.22: wM-Bus Gateway and Meter memory usage for three different messages at different length

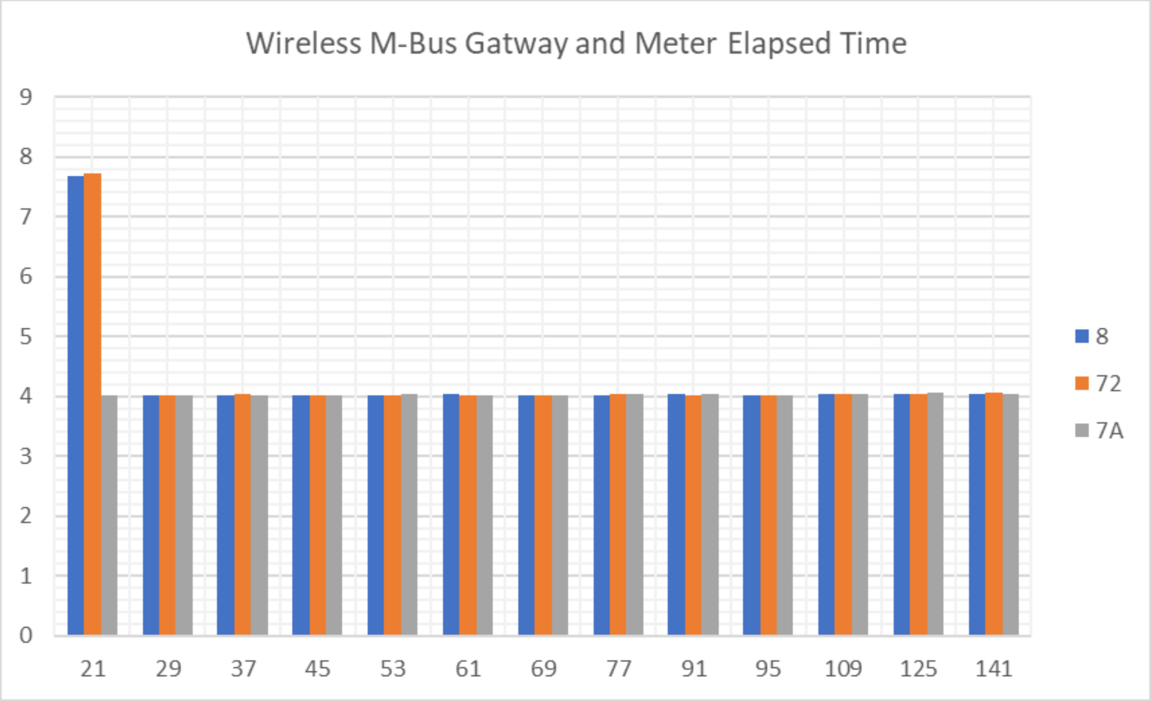
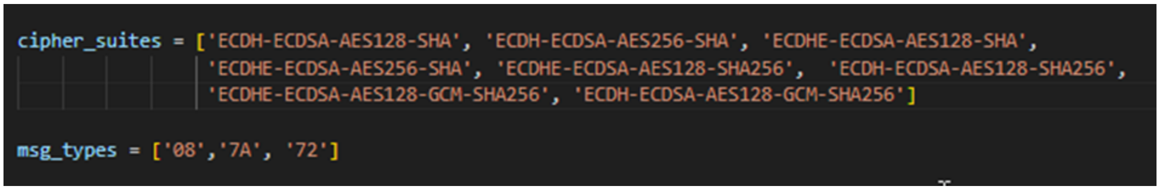


Figure 5.23: wM-Bus Gateway and Meter elapsed time for three different messages at different length

5.4 Phase Three: wM-Bus with TLS 1.2

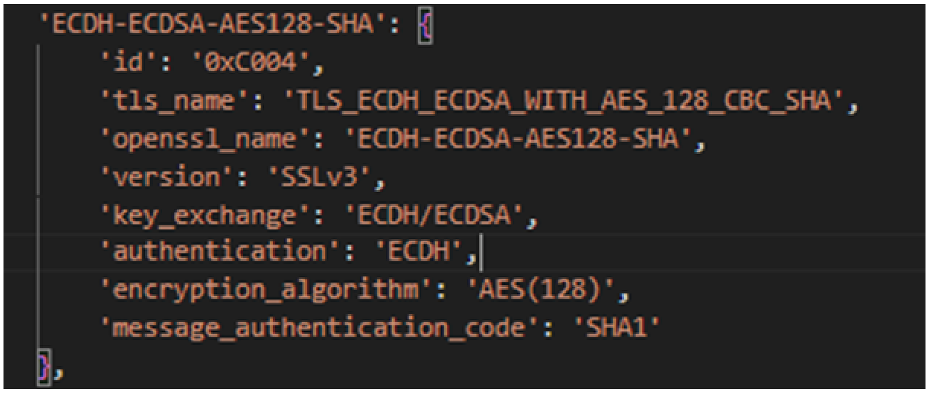
In this phase of the experiment, the focus shifted towards fortifying the wM-Bus communication protocol by integrating a robust TLS (Transport Layer Security) secure layer. Building upon the foundation laid in the preceding stage, which involved the development of Python code for wM-Bus communication without any security measures, this subsequent step aimed at enhancing the protocol's resilience through the implementation of TLS. The implementation leverages key Python libraries, including socket and cryptography, to employ essential cryptographic functionalities such as HMAC (Hash-based Message Authentication Code), cipher operations for encryption and decryption, as well as hashing and HKDF (HMAC-based Extract-and-Expand Key Derivation Function). These libraries provide the necessary tools to establish a secure communication channel between the wM-Bus Meter (Client) and Gateway(Server). To align the security measures with the peculiarities of the wM-Bus frame size, specific cipher suites were selected. These suites, such as 'ECDHE-ECDSA-AES256-GCM-SHA384,' 'ECDHE-RSA-AES256-GCM-SHA384,' and others presented in Figure (5.24,5.24) were chosen to ensure compatibility with the wM-Bus frame size. Notably, diverse key exchange mechanisms, including X448 and



```
cipher_suites = ['ECDH-ECDSA-AES128-SHA', 'ECDH-ECDSA-AES256-SHA', 'ECDHE-ECDSA-AES128-SHA',
                 'ECDHE-ECDSA-AES256-SHA', 'ECDHE-ECDSA-AES128-SHA256',  'ECDH-ECDSA-AES128-SHA256',
                 'ECDHE-ECDSA-AES128-GCM-SHA256', 'ECDH-ECDSA-AES128-GCM-SHA256']

msg_types = ['08','7A', '72']
```

Figure 5.24: TLS cipher suite used for wM-Bus implementation

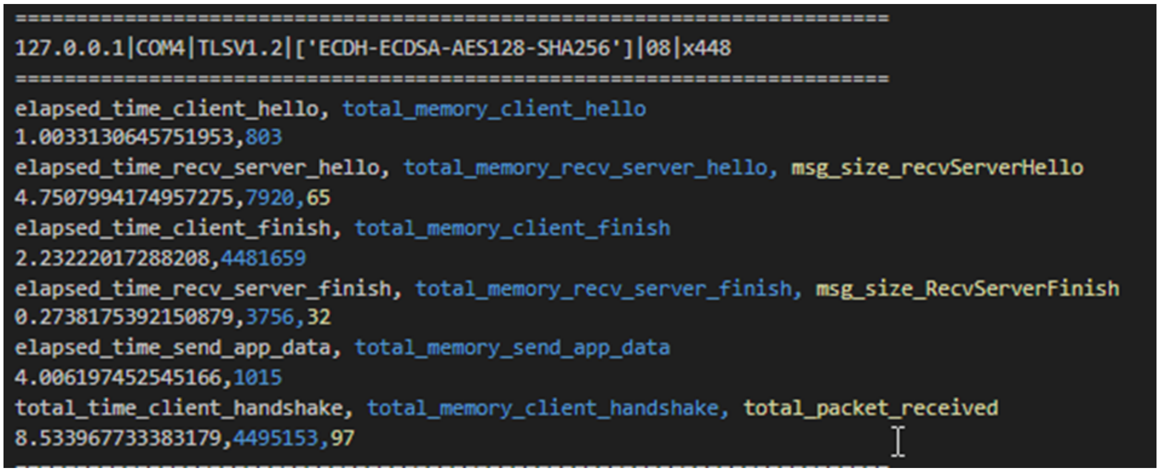


```
'ECDH-ECDSA-AES128-SHA': {
    'id': '0xC004',
    'tls_name': 'TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA',
    'openssl_name': 'ECDH-ECDSA-AES128-SHA',
    'version': 'SSLv3',
    'key_exchange': 'ECDH/ECDSA',
    'authentication': 'ECDH',
    'encryption_algorithm': 'AES(128)',
    'message_authentication_code': 'SHA1'
},
```

Figure 5.25: One example of a TLS Cipher suite: ’EDCDH-ECDSA-AES128-SHA’

X25519, were integrated with each cipher suite. This comprehensive approach allows for a thorough examination of the security properties associated with each combination. The experiment involved running tests for each selected cipher suite and key exchange method to evaluate their impact on the security parameters of the wM-Bus communication. By delving into the intricacies of TLS and cryptographic operations, this phase of the experiment aims to provide a robust and secure framework for wM-Bus communication, addressing potential vulnerabilities and fortifying the protocol against unauthorized access or data breaches. During the execution of the developed Python code for implementing a TLS secure layer in the wM-Bus communication protocol, a meticulous approach is taken to record and analyze the outcomes. The code execution logs are systematically saved in separate log files for each cipher suite utilized in the experiment. This logging mechanism is crucial for tracking and assessing the performance and security properties associated with each configuration. The output for each cipher suite provides insightful information regarding the handshake process, encompassing crucial steps such as ’Client Hello,’ ’Server Hello,’ ’Client Finish,’ ’Server Finish,’ and the transmission of application data. Each iteration within a specific cipher suite is logged comprehensively, detailing the security properties, memory usage, and elapsed time for each phase of the communication.

Specifically, for the wM-Bus protocol with a designated message type ’08,’ the mem-



```
========================================================================
127.0.0.1|COM4|TLSV1.2|['ECDH-ECDSA-AES128-SHA256']|08|x448
========================================================================
elapsed_time_client_hello, total_memory_client_hello
1.0033130645751953,803
elapsed_time_recv_server_hello, total_memory_recv_server_hello, msg_size_recvServerHello
4.7507994174957275,7920,65
elapsed_time_client_finish, total_memory_client_finish
2.23222017288208,4481659
elapsed_time_recv_server_finish, total_memory_recv_server_finish, msg_size_RecvServerFinish
0.2738175392150879,3756,32
elapsed_time_send_app_data, total_memory_send_app_data
4.006197452545166,1015
total_time_client_handshake, total_memory_client_handshake, total_packet_received
8.533967733383179,4495153,97
```

Figure 5.26: Example of the log file for one Cipher suite at the Meter side.

ory usage and time elapsed metrics are calculated meticulously for various stages of the TLS handshake. These stages include the initial exchange of greetings ('Client Hello' and 'Server Hello'), the finalization of the connection ('Client Finish' and 'Server Finish'), and the transmission of application data as depicted in Figure (5.26). This approach enables a granular examination of the impact of each cipher suite on the security and performance aspects of the wM-Bus communication. The log files serve as a valuable resource for post-execution analysis, allowing researchers and developers to identify trends, optimize configurations, and ensure the robustness of the TLS integration within the wM-Bus protocol. We conducted a comprehensive evaluation of the performance characteristics of various cipher suites, as applied to both Meter and Gateway systems, the results of which are detailed in Tables 5.7 and 5.8, respectively. To provide an integrated view, we also synthesized the cumulative performance outcomes for both the Gateway and Meter in Table 5.9. This consolidated analysis was aimed at identifying the most efficient or suitable cipher suite for deployment in specific environments or applications, with a particular focus on assessing efficiency through key indicators such as processing speed (time) and the consumption of system resources (memory usage). The following three sections will have a deeper insight of these tables, offering an overview and a detailed analysis of the results to further elucidate the implications of our findings.

5.4.1 wM-Bus with TLS - Meter Analysis

To analyze the Table closely 5.9, as the data in terms of the performance metrics listed for two key exchange algorithms x448 and x25519. The metrics provided include the time

Table 5.7: Wireless M-Bus with TLS Handshake Memory Usage and Elapsed Time For the Meter (Client)

			Key Exchange				
			x448		**x25519**		
No	**Cipher**	**Msg_Type**	**Time**	**Mem_Usage**	**Time**	**Mem_Usage**	**Size**
1	*ECDH-ECDSA-AES128-SHA*	08	8.427475214	4485427	8.489712	4486982	97
2	*ECDH-ECDSA-AES256-SHA*	08	8.176381826	4487073	8.391309	4486768	97
3	*ECDHE-ECDSA-AES128-SHA*	08	8.564642668	4486847	8.494767	4486852	97
4	*ECDHE-ECDSA-AES256-SHA*	08	8.547471762	4486827	8.552336	4487077	97
5	*ECDHE-ECDSA-AES128-SHA256*	08	8.583416224	4486678	8.741161	4486962	97
6	*ECDH-ECDSA-AES128-SHA256*	08	8.538057566	4486719	8.555192	4486724	97
7	*ECDHE-ECDSA-AES128-GCM-SHA256*	08	8.216920137	4486343	8.121013	4486525	97
8	*ECDH-ECDSA-AES128-GCM-SHA256*	08	8.129461288	4486222	8.12194	4486226	97
9	*ECDH-ECDSA-AES128-SHA*	7A	8.591912746	4487030	8.393756	4486799	97
10	*ECDH-ECDSA-AES256-SHA*	7A	8.465797424	4486828	8.576565	4486832	97
11	*ECDHE-ECDSA-AES128-SHA*	7A	8.380492449	4487030	8.421456	4487097	97
12	*ECDHE-ECDSA-AES256-SHA*	7A	8.343997717	4486828	8.452907	4486768	97
13	*ECDHE-ECDSA-AES128-SHA256*	7A	8.556341648	4486657	8.661117	4486906	97
14	*ECDH-ECDSA-AES128-SHA256*	7A	8.745227098	4486719	8.593612	4486906	97
15	*ECDHE-ECDSA-AES128-GCM-SHA256*	7A	8.179555893	4486343	8.119965	4486292	97
16	*ECDH-ECDSA-AES128-GCM-SHA256*	7A	8.157888412	4486288	8.081982	4486226	97
17	*ECDH-ECDSA-AES128-SHA*	72	9.134933233	4486844	8.336077	4487098	97
18	*ECDH-ECDSA-AES256-SHA*	72	8.417791367	4486764	9.325000	4486768	97
19	*ECDHE-ECDSA-AES128-SHA*	72	8.398283482	4486784	8.477798	4486789	97
20	*ECDHE-ECDSA-AES256-SHA*	72	8.412582397	4486764	8.270789	4486831	97
21	*ECDHE-ECDSA-AES128-SHA256*	72	8.700239897	4486719	8.686308	4486723	97
22	*ECDH-ECDSA-AES128-SHA256*	72	8.700618267	4486720	8.717626	4486906	97
23	*ECDHE-ECDSA-AES128-GCM-SHA256*	72	8.064455748	4486165	8.155625	4486580	97
24	*ECDH-ECDSA-AES128-GCM-SHA256*	72	8.117798328	4486222	8.43094	4486454	97

taken and memory usage for various ciphers, along with a size measurement which seems to be constant across all entries. We can observe that the time taken for key exchange operations varies across different ciphers for both x448 and x25519. It appears that x25519 generally has slightly lower time values compared to x448, indicating faster performance in these particular data samples. The memory usage is also varying between the ciphers for both x448 and x25519. There doesn't seem to be a consistent trend indicating which of the two key exchange algorithms consistently uses less memory. The table includes both ECDH and ECDHE ciphers with AES128 and AES256, some with SHA and others with SHA256. There are three distinct message types 08, 7A, and 72. There does not seem to be a consistent pattern of performance within these message types. The packet size field is constant at 97 for all entries, which represent a fixed block size. Consistency across key exchange algorithms, Some ciphers show more variation in time and memory usage between x448 and x25519 than others. For instance, ECDHE-ECDSA-AES128-GCM-SHA256 (No. 23) shows a significant difference in time performance between x448 and x25519. To conclude, this table suggests that x25519 may be a slightly faster algorithm for key exchange in terms of time taken, but the difference in memory usage between x448 and x25519.

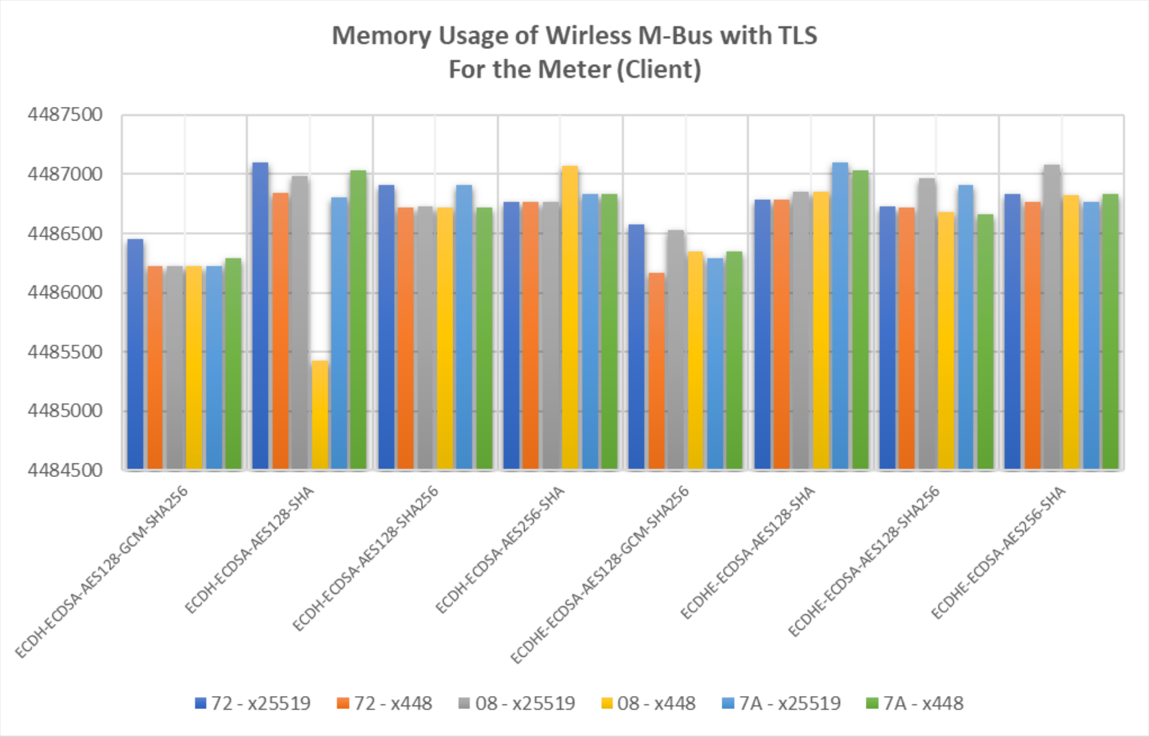


Figure 5.27: wM-Bus with TLS shows handshake memory usage for the meter with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges.

The memory usage seems to be quite stable across different tests as depicted in Figures 5.27 and 5.28, hovering around the 4484500 to 4487500 bytes range. There is not a significant difference between the various configurations, which might suggest that the memory overhead for different TLS configurations in the context of wM-Bus is relatively consistent, at least within the range of configurations tested here.

Based on the elapsed time in Figure 5.29 and 5.30 for wM-Bus with TLS security layer with the combination of two key exchange “X25519” and “x448” along with three different wM-Bus message type (72,7A, 08). The message type ”72” are generally the highest across the different cipher/key exchange combinations, which might suggest that this message type takes the longest time to process. The ”x448” key exchange mechanism seems to result in a longer elapsed time compared to ”x25519”, which might suggest that ”x448” is more computationally intensive or secure, hence the longer processing time. The ”ECDH-ECDSA-AES256-SHA” cipher suite combination tends to have higher elapsed times across all message types compared to ”ECDH-ECDSA-AES128-GCM-SHA256”, possibly indicating that the increased security from using a 256-bit key comes with a performance

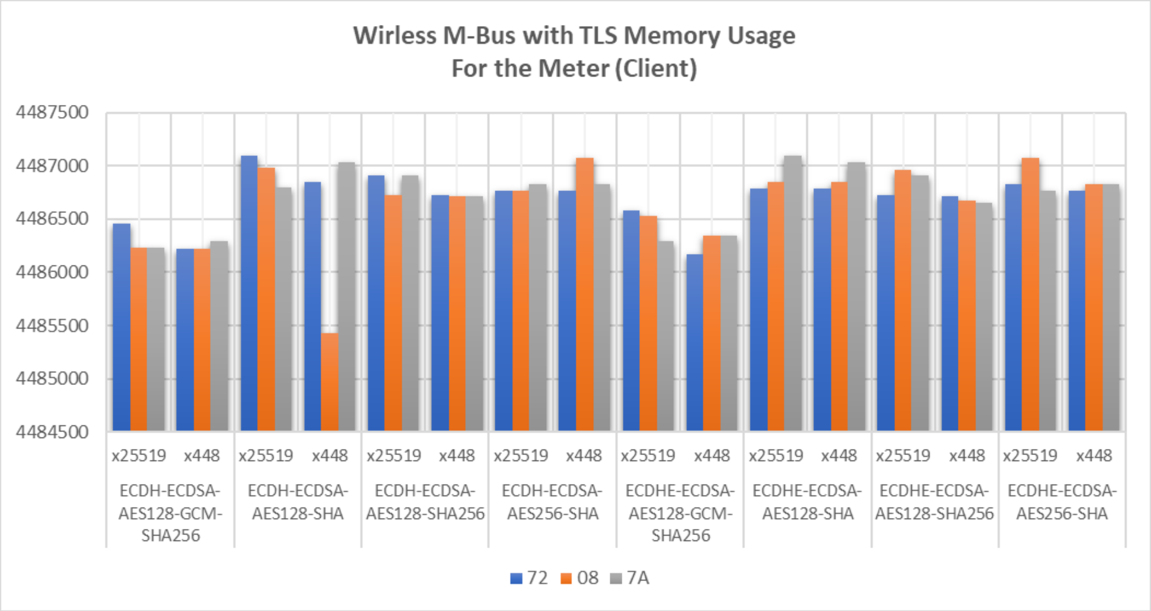


Figure 5.28: wM-Bus with TLS showing handshake memory usage for the meter with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges

cost. The quickest handshakes appear to be associated with the ”x25519” curve for key exchange, particularly with ”AES128-GCM-SHA256” cipher. Overall, the chart suggests that the choice of cryptographic algorithms can significantly impact the efficiency of the TLS handshake process in a wM-Bus system from the client’s perspective.

5.4.2 wM-Bus with TLS - Gateway Analysis

Similarly, this Table 5.8 presents as well the times for key exchanges using x448 and x25519 algorithms are fairly close, with some variations. However, the absolute times are higher across all entries compared to the meter side, which could be due to the server handling additional processing tasks. As for the Memory usage seems consistent across both algorithms, with slight variations between entries. Both x448 and x25519 algorithms show similar memory usage, indicating that from a memory standpoint, either algorithm could be suitable for the server side. Unlike the previous table, the size now varies between 184, 201, and 217. This suggests different configurations or requirements for the key exchange process on the server side compared to the meter side. It’s notable that the size does not correlate directly with time or memory usage, which implies that the size may not be a determining factor in the performance metrics. The ciphers listed are the same types as before (ECDH, ECDHE, AES128, AES256, SHA, SHA256), but the performance on the

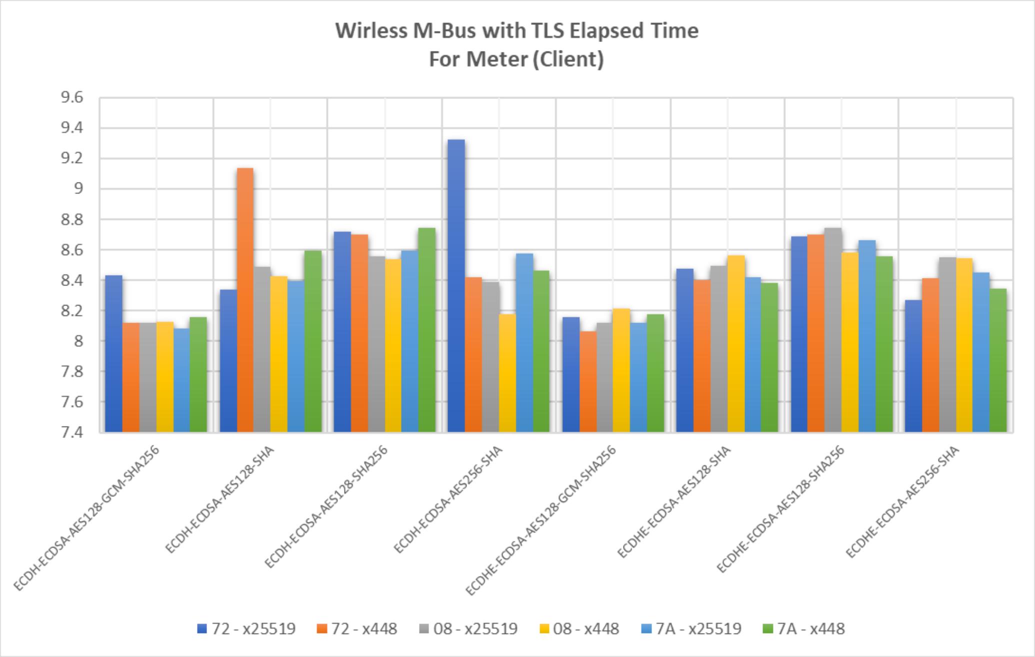


Figure 5.29: wM-Bus with TLS showing handshake elapsed time usage for the meter with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges

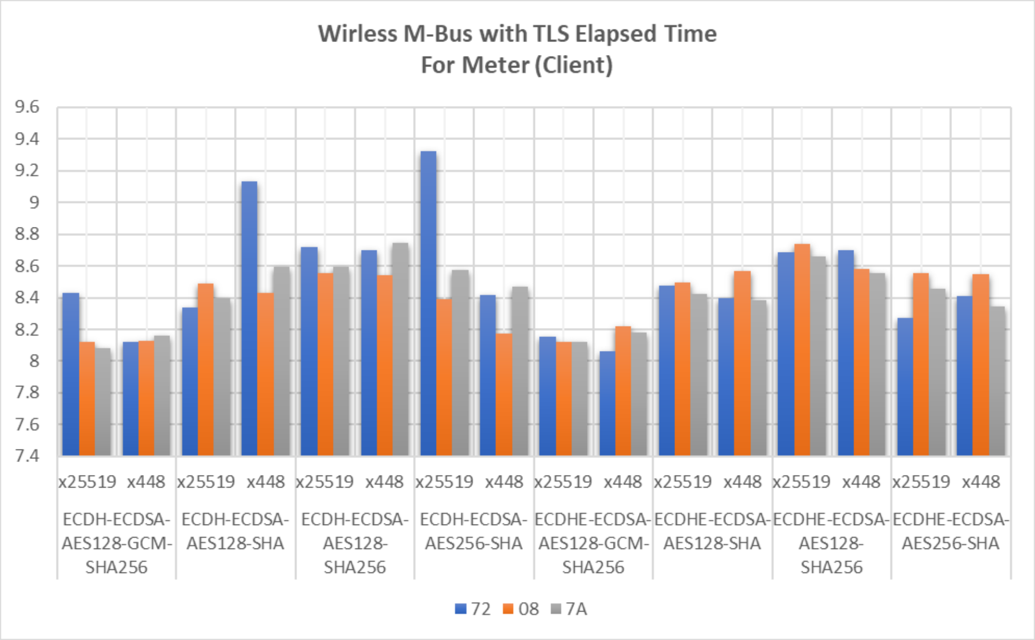


Figure 5.30: wM-Bus with TLS showing memory usage for the meter with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges

Table 5.8: Wireless M-Bus with TLS Handshake Memory Usage and Elapsed Time For the Gateway (Server)

			Key Exchange					
			x448			**x25519**		
No	**Cipher**	**Msg_Type**	**Time**	**Mem_Usage**	**Size**	**Time**	**Mem_Usage**	**Size**
1	*ECDH-ECDSA-AES128-SHA*	08	11.46119595	4496224	201	11.55862	4487194	201
2	*ECDH-ECDSA-AES256-SHA*	08	11.44489241	4487274	201	11.42703	4487091	201
3	*ECDHE-ECDSA-AES128-SHA*	08	11.52463198	4487284	201	11.56238	4487194	201
4	*ECDHE-ECDSA-AES256-SHA*	08	11.63676596	4487316	201	11.51599	4487373	201
5	*ECDHE-ECDSA-AES128-SHA256*	08	11.78119159	4486927	217	11.80291	4487213	217
6	*ECDH-ECDSA-AES128-SHA256*	08	11.67678905	4487209	217	11.73468	4487065	217
7	*ECDHE-ECDSA-AES128-GCM-SHA256*	08	11.30930567	4486841	184	11.24896	4486848	184
8	*ECDH-ECDSA-AES128-GCM-SHA256*	08	11.27959967	4486719	184	11.21692	4486845	184
9	*ECDH-ECDSA-AES128-SHA*	7A	11.60709810	4487242	201	11.51013	4487246	201
10	*ECDH-ECDSA-AES256-SHA*	7A	11.51596951	4487274	201	11.52751	4487225	201
11	*ECDHE-ECDSA-AES128-SHA*	7A	11.54482150	4487242	201	11.50578	4487246	201
12	*ECDHE-ECDSA-AES256-SHA*	7A	11.47712731	4487274	201	11.48832	4487373	201
13	*ECDHE-ECDSA-AES128-SHA256*	7A	11.71333551	4486872	217	11.74659	4486815	217
14	*ECDH-ECDSA-AES128-SHA256*	7A	11.74167538	4487061	217	11.71248	4487065	217
15	*ECDHE-ECDSA-AES128-GCM-SHA256*	7A	11.23067570	4486664	184	11.22002	4486605	184
16	*ECDH-ECDSA-AES128-GCM-SHA256*	7A	11.24217653	4486753	184	11.25976	4486845	184
17	*ECDH-ECDSA-AES128-SHA*	72	11.64042640	4487242	201	11.73192	4487246	201
18	*ECDH-ECDSA-AES256-SHA*	72	11.55647278	4487369	201	11.60641	4487225	201
19	*ECDHE-ECDSA-AES128-SHA*	72	11.57512879	4487242	201	11.55597	4487057	201
20	*ECDHE-ECDSA-AES256-SHA*	72	11.59541583	4487087	201	11.49123	4487278	201
21	*ECDHE-ECDSA-AES128-SHA256*	72	11.78355789	4487061	217	11.75386	4487065	217
22	*ECDH-ECDSA-AES128-SHA256*	72	11.81813121	4487061	217	11.79874	4487213	217
23	*ECDHE-ECDSA-AES128-GCM-SHA256*	72	11.28909016	4486719	184	11.28061	4486759	184
24	*ECDH-ECDSA-AES128-GCM-SHA256*	72	11.32802105	4486808	184	11.51248	4486847	184

gateway might differ due to the different operational environment and tasks it's handling. he consistency in performance between x448 and x25519 is observed here as well, with neither algorithm consistently outperforming the other in the server context. On the gateway (server side), both x448 and x25519 algorithms appear to perform similarly in terms of time and memory usage.

At the server side looking into the memory usage as presented in Figures 5.31 and 5.32, the ”x25519-72” configurations, across different cipher suites, show varying memory usage but are generally in the middle range of the memory usage spectrum. The ”x25519-08” and ”x448-72” configurations also show a range of memory usage but do not consistently occupy the highest or lowest memory usage. The ”x448-08” configurations tend to have the lower memory usage across the cipher suites compared to ”x448-7A”.

In addition, the Elapsed time at the Gateway (server) side presented in Figures 5.33 and 5.34, using different cipher suites and key exchange algorithms (x25519 and x448). In almost all cases, the x25519 algorithm shows a slightly lower elapsed time compared to x448, suggesting it is faster for these operations. The pattern of x25519 being faster than x448 is consistent across different message types (72, 08, and 7A). Some cipher suites, such as ECDH-ECDSA-AES128-SHA and ECDH-ECDSA-AES256-SHA, show

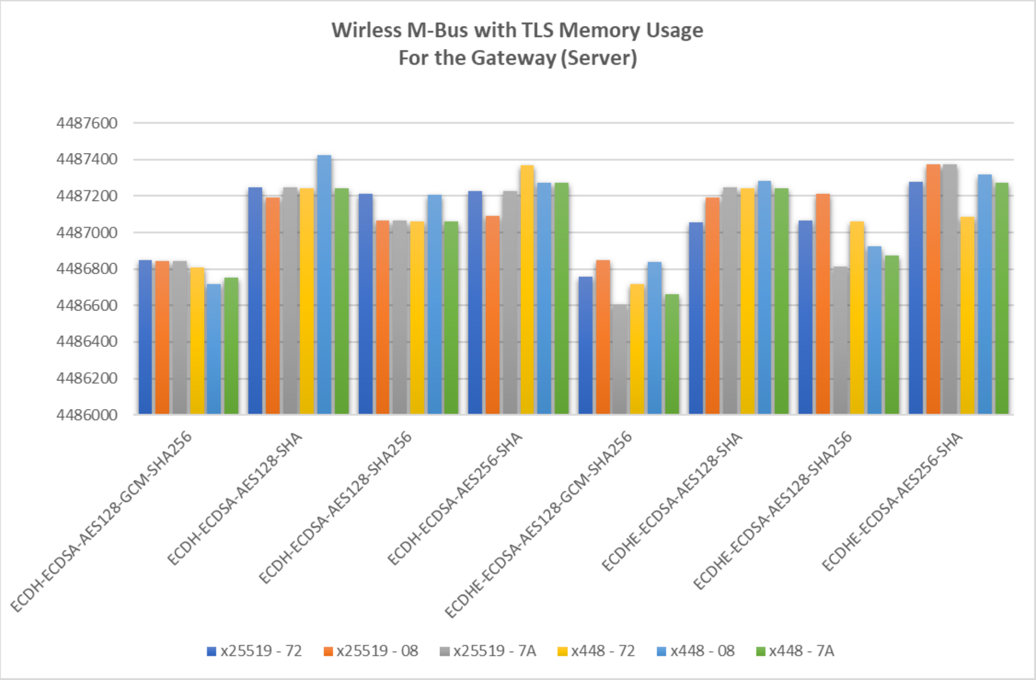


Figure 5.31: wM-Bus with TLS showing handshake Memory Usage for the Gateway with the combination of 8 different cipher suites, 3 different wM-Bus messages and two different key exchanges

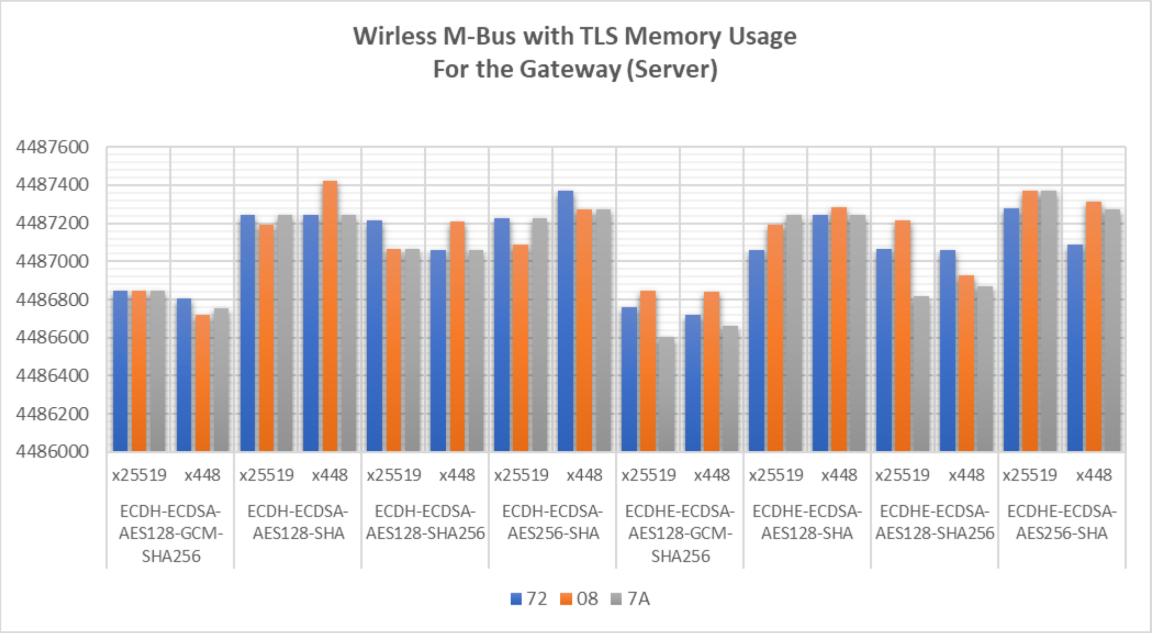


Figure 5.32: wM-Bus with TLS showing memory usage for the Gateway with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges

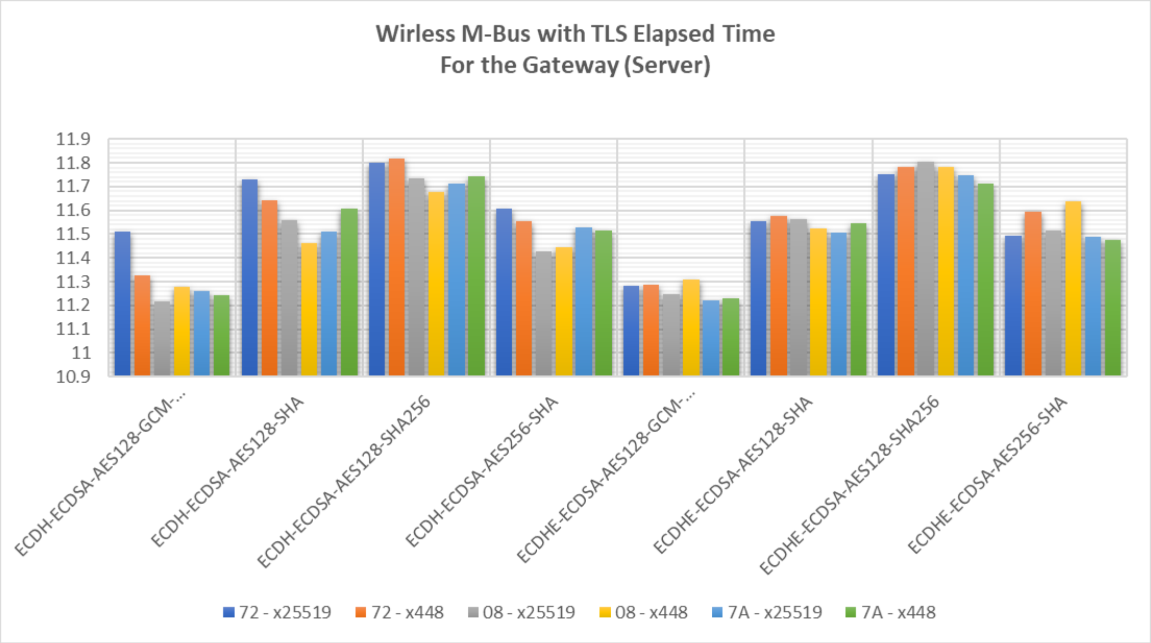


Figure 5.33: wM-Bus with TLS showing handshake elapsed time for the Gateway with the combination of 8 different cipher suites, 3 different wM-Bus messages and two different key exchanges

relatively small differences in elapsed time between the two key exchange algorithms. In contrast, ECDHE-ECDSA-AES128-SHA256 shows a more noticeable difference. Consistently, x25519 demonstrates better performance (lower elapsed time) across the various cipher suites and message types on the gateway side. The elapsed times for message types 72 and 08 are generally higher than for message type 7A. Based on the analysis of the gateway side of wM-Bus communications with TLS, the x25519 key exchange algorithm seems to offer slightly better performance in terms of elapsed time compared to x448, across various cipher suites and message types. This consistent performance edge could be a factor in selecting x25519 for efficiency purposes in this specific context.

5.4.3 wM-Bus with TLS - Handshake Meter & Gateway Combined Analysis

This section presents an overall overview of the handshake process between the meter and gateway combined, as detailed in the provided in the Table 5.9 and the figures for overall memory usage as depicted in the figures 5.35 and 5.36. The x25519 also appears to use slightly less memory than x448 in most entries, but again, the difference is not substantial as shown in Figure 5.37 and 5.38. Looking into the elapsed time the x25519 sometimes

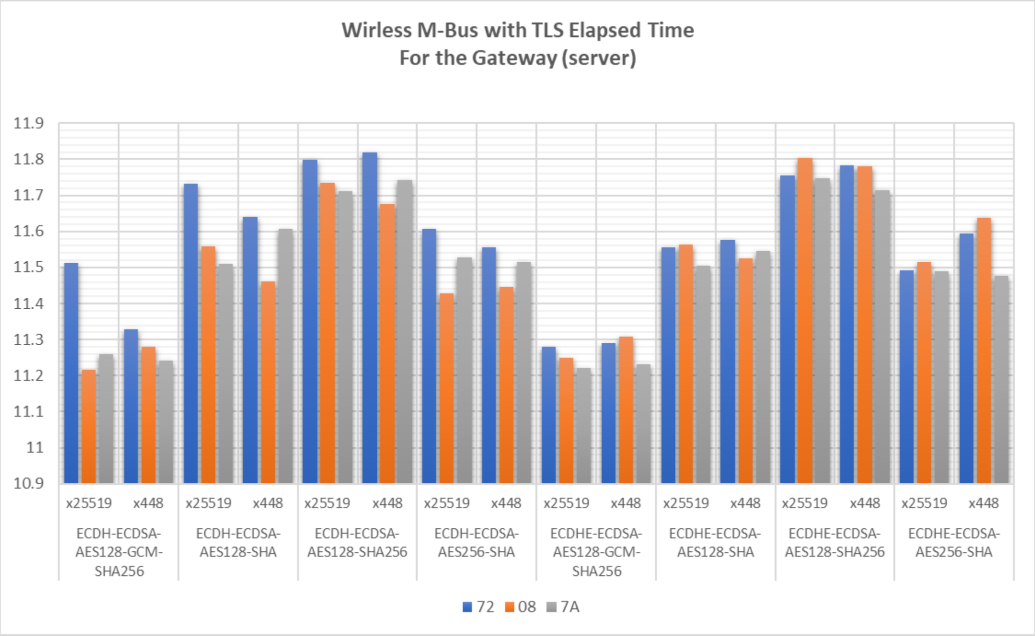


Figure 5.34: wM-Bus with TLS showing elapsed time for the Gateway with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges

performs faster than x448, but the differences are minimal, suggesting similar performance characteristics for key exchanges. Finally the packet size seems to be consistent for each cipher suite across both algorithms as shown in Figure 5.39 and 5.40. However, the packet sizes are all within a close range, approximately between 270 to 320 bytes. The x448 key exchange tends to result in slightly larger packet sizes across all cipher suites and message types compared to x25519. The ECDH-ECDSA-AES256-SHA cipher suite generally results in the largest packet sizes, which is expected as AES256 uses a larger key than AES128, potentially leading to larger handshake messages. Currently, there does not appear to be a significant variance in packet size based on M-Bus message type alone, as the variation is more influenced by the key exchange algorithm and the cipher suite. The “72” message type consistently has the smallest packet size across different key exchanges and cipher suites, while 7A often has the largest. Some of the implication that if minimizing packet size is crucial, for instance in bandwidth-constrained environments, the x25519 key exchange with AES128 might be preferable. Larger packet sizes may be justified by the increased security provided by AES256 or the x448 key exchange, but the specific security needs would have to be weighed against the cost of increased data transmission size.

Table 5.9: Wireless M-Bus with TLS Handshake Memory Usage and Elapsed Time For both Gateway and Meter

			Key Exchange					
			x448			**x25519**		
No	**Cipher**	**Msg_Type**	**Time**	**Mem_Usage**	**Size**	**Time**	**Mem_Usage**	**Size**
1	*ECDH-ECDSA-AES128-SHA*	08	19.88867116	8981651	298	20.04833	8974176	298
2	*ECDH-ECDSA-AES256-SHA*	08	19.62127423	8974347	298	19.81834	8973859	298
3	*ECDHE-ECDSA-AES128-SHA*	08	20.08927464	8974131	298	20.05715	8974046	298
4	*ECDHE-ECDSA-AES256-SHA*	08	20.18423772	8974143	298	20.06833	8974450	298
5	*ECDHE-ECDSA-AES128-SHA256*	08	20.36460781	8973605	314	20.54407	8974175	314
6	*ECDH-ECDSA-AES128-SHA256*	08	20.21484661	8973928	314	20.28987	8973789	314
7	*ECDHE-ECDSA-AES128-GCM-SHA256*	08	19.52622581	8973184	281	19.36998	8973373	281
8	*ECDH-ECDSA-AES128-GCM-SHA256*	08	19.40906096	8972941	281	19.33886	8973071	281
9	*ECDH-ECDSA-AES128-SHA*	7A	20.19901085	8974272	298	19.90388	8974045	298
10	*ECDH-ECDSA-AES256-SHA*	7A	19.98176694	8974102	298	20.10407	8974057	298
11	*ECDHE-ECDSA-AES128-SHA*	7A	19.92531395	8974272	298	19.92724	8974343	298
12	*ECDHE-ECDSA-AES256-SHA*	7A	19.82112503	8974102	298	19.94123	8974141	298
13	*ECDHE-ECDSA-AES128-SHA256*	7A	20.26967716	8973529	314	20.40771	8973721	314
14	*ECDH-ECDSA-AES128-SHA256*	7A	20.48690248	8973780	314	20.30610	8973971	314
15	*ECDHE-ECDSA-AES128-GCM-SHA256*	7A	19.41023159	8973007	281	19.33999	8972897	281
16	*ECDH-ECDSA-AES128-GCM-SHA256*	7A	19.40006495	8973041	281	19.34175	8973071	281
17	*ECDH-ECDSA-AES128-SHA*	72	20.77535963	8974086	298	20.06800	8974344	298
18	*ECDH-ECDSA-AES256-SHA*	72	19.97426414	8974133	298	20.93141	8973993	298
19	*ECDHE-ECDSA-AES128-SHA*	72	19.97341228	8974026	298	20.03377	8973846	298
20	*ECDHE-ECDSA-AES256-SHA*	72	20.00799823	8973851	298	19.76202	8974109	298
21	*ECDHE-ECDSA-AES128-SHA256*	72	20.48379779	8973780	314	20.44017	8973788	314
22	*ECDH-ECDSA-AES128-SHA256*	72	20.51874948	8973781	314	20.51636	8974119	314
23	*ECDHE-ECDSA-AES128-GCM-SHA256*	72	19.35354590	8972884	281	19.43624	8973339	281
24	*ECDH-ECDSA-AES128-GCM-SHA256*	72	19.44581938	8973030	281	19.94342	8973301	281

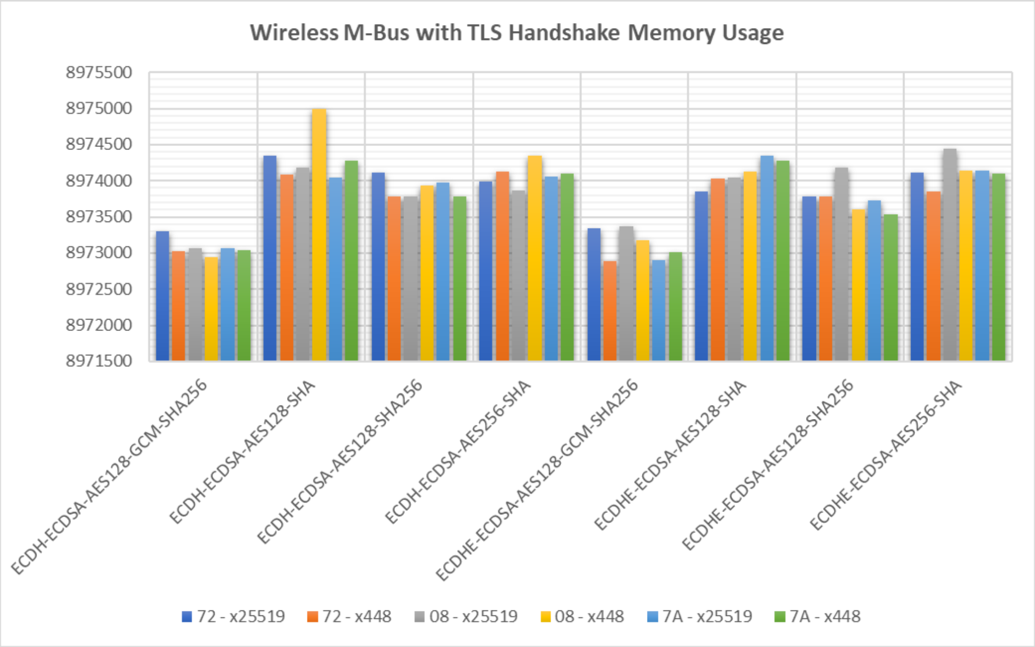


Figure 5.35: wM-Bus with TLS showing handshake Memory Usage with the combination of 8 different cipher suites, 3 different wM-Bus messages and two different key exchanges

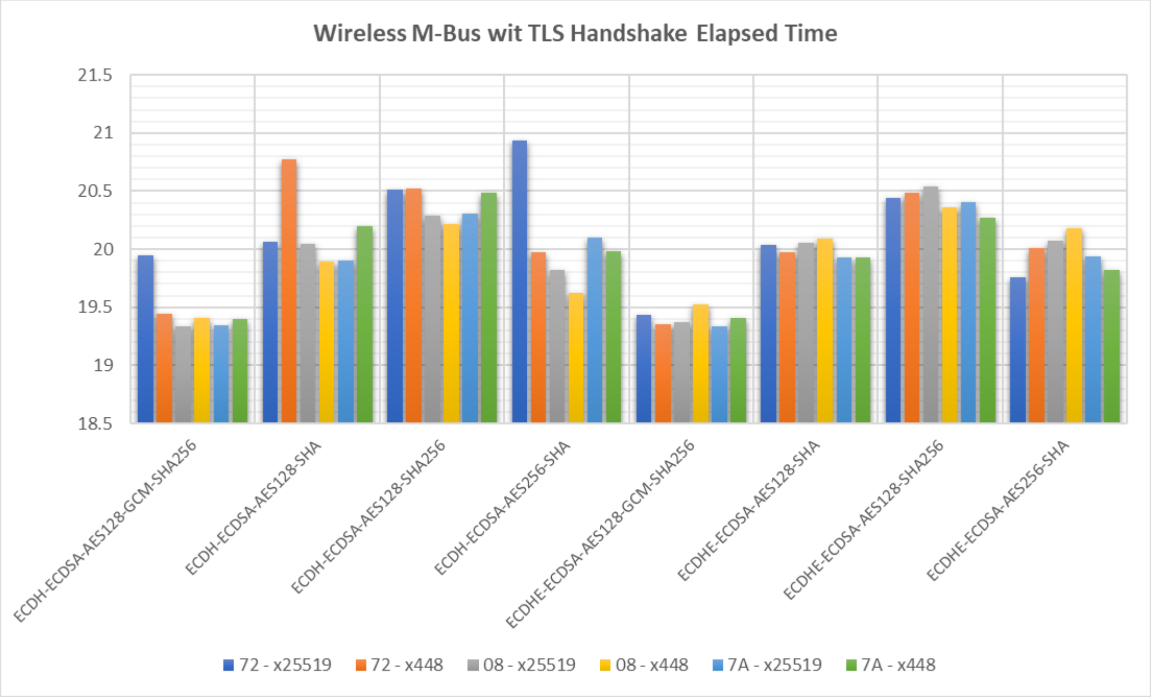


Figure 5.36: wM-Bus with TLS showing Memory Usage with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges

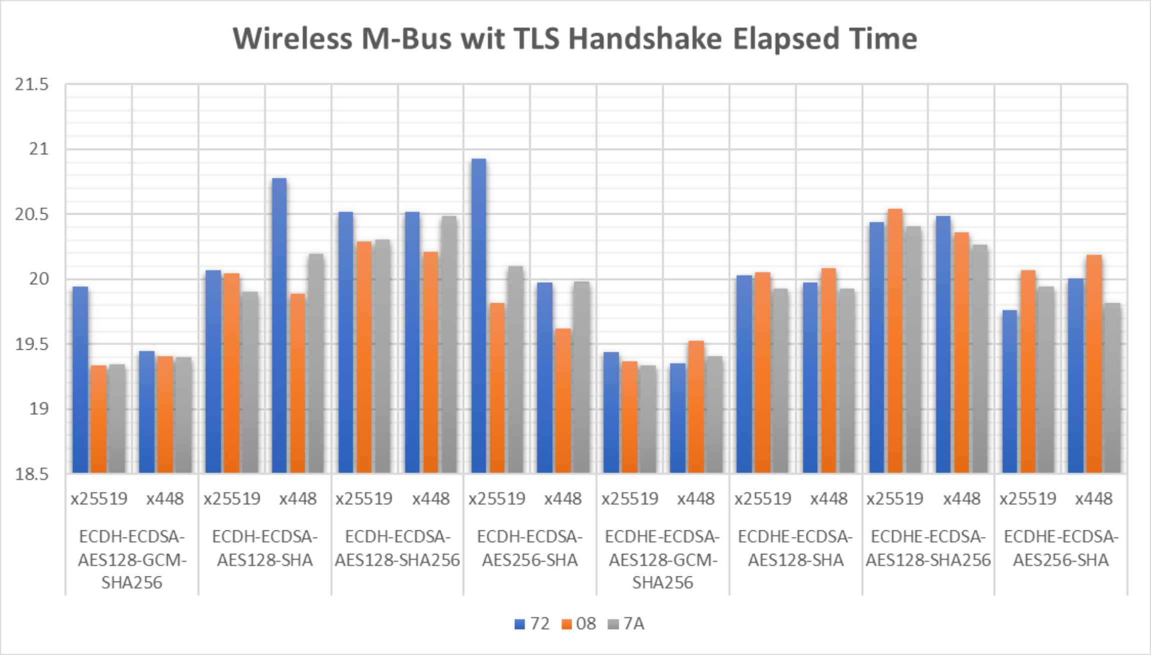


Figure 5.37: wM-Bus with TLS showing handshake Elapsed Time with the combination of 8 different cipher suites, 3 different wM-Bus messages and two different key exchanges

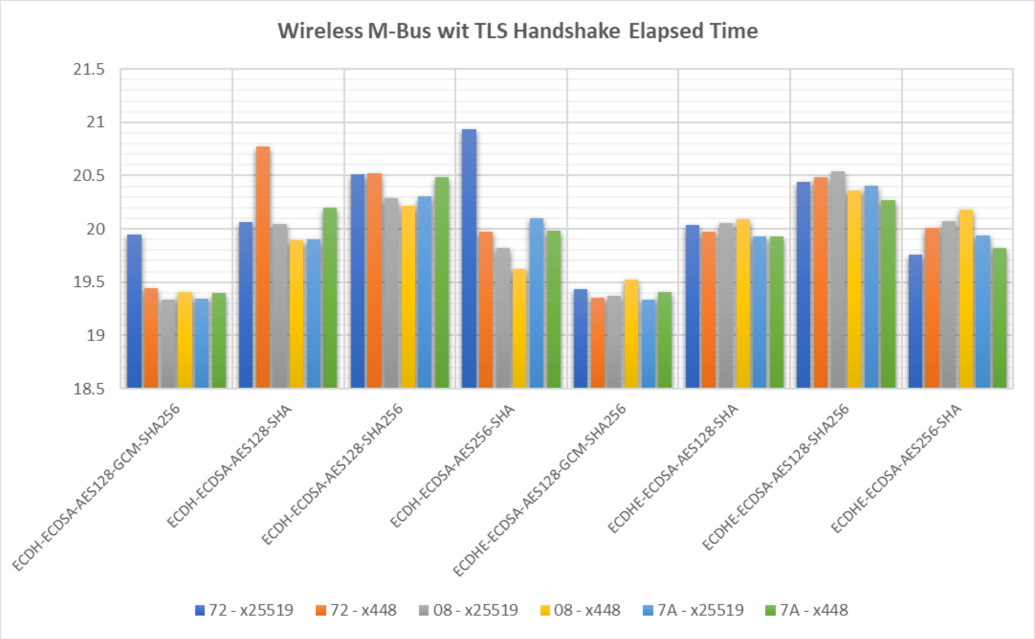


Figure 5.38: wM-Bus with TLS showing Elapsed Time with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges

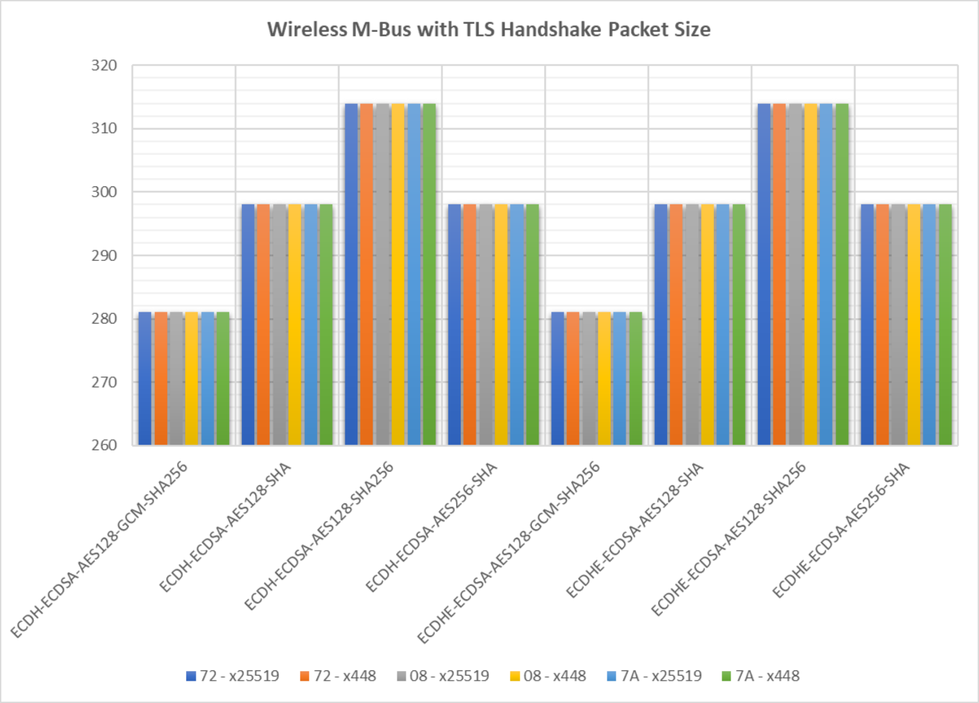


Figure 5.39: wM-Bus with TLS showing handshake Elapsed Time with the combination of 8 different cipher suites, 3 different wM-Bus messages and two different key exchanges

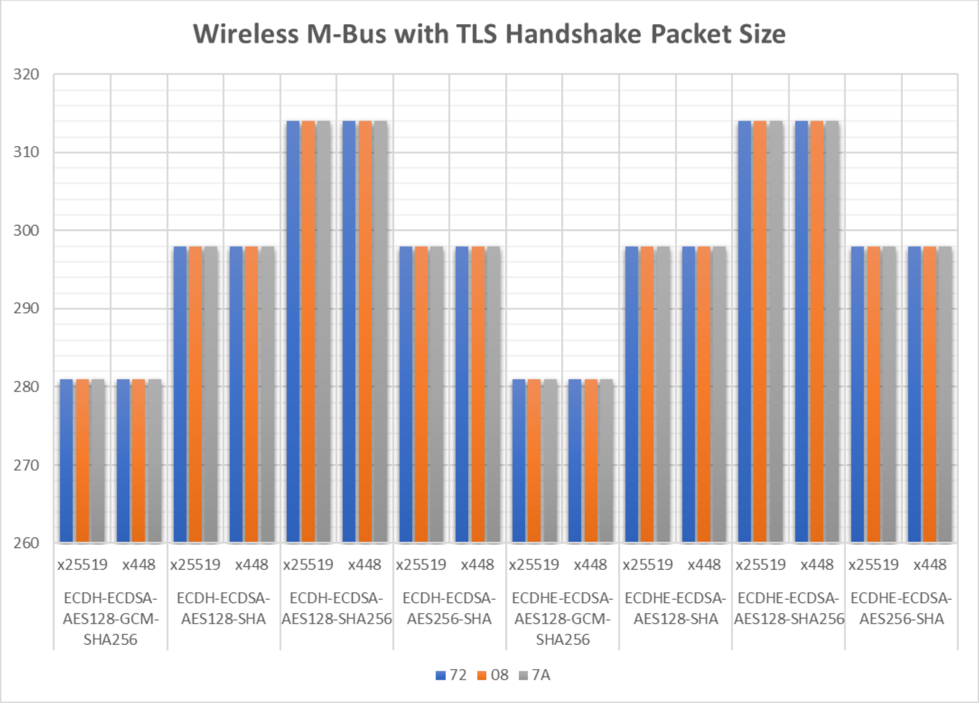


Figure 5.40: wM-Bus with TLS showing Elapsed Time with the combination of 8 different cipher suite, 3 different wM-Bus messages and two different key exchanges

5.5 Phase Four: wM-Bus with NPF

In the fourth phase of our project, we embarked on the development of wM-Bus technology, integrating it seamlessly with NPF. This integration allowed us to implement and evaluate six distinct predefined communication patterns: IX, KN, NK, NX, and XX, all within the context of the NPF. To accomplish this, we leveraged the power and flexibility of Python, developing code that made use of both the wireless and NPF libraries. One of the primary objectives of this phase was to prioritize the security of communication between the meter and the gateway. To this end, we utilize the NPF to strengthen the confidentiality and integrity of the data being transmitted. By doing so, we aimed to ensure that sensitive information remained protected from potential threats. In order to estimate the effectiveness and efficiency of our implementation, we employed the same metrics throughout our testing process. Specifically, we meticulously measured memory usage and the time required for the handshake process as depicted in Table 9 for the gateway, Table 10 for the meter and Table 11 reflect both Meter and Gateway as the entire handshake and sending encrypted application data. This consistent evaluation allowed us to gain insights into the performance and resource demands of our solution, ultimately aiding us in fine-tuning and optimizing our system for real-world deployment.

Table 5.10: Wireless M-Bus with Noise Framework Handshake Calculations – Gateway

Pattern Name	Memory Recv Client Hello	Memory ServerHello	Memory Recv ClientFinish	Memory Recv AppData	Time Recv ClientHello	Time ServerHello	Time Recv ClientFinish	Time Recv AppData
IX	153299	6458	0	1692	7.155589581	4.0105300	0	3.1310685
KN	145683	3042	0	1564	3.904595852	4.0055332	0	2.8832216
KX	144679	3451	0	1420	3.904593229	4.0085280	0	3.1200774
NK	145056	2594	0	1300	3.985543966	4.0045314	0	2.8822234
NX	144286	2955	0	1188	3.917584658	4.0065310	0	3.1220775
XX	14612	5067	3651	1647	4.166036606	4.0182130	3.16847253	3.9567633

Table 5.11: Wireless M-Bus with Noise Framework Handshake Calculations – Meter

Pattern	Memory Init	Memory ClientHello	Memory Recv ServerHello	Memory ClientFinish	Memory AppData	Time Init	Time Client Hello	Time Recv ServerHello	Time ClientFinish	Time Recv AppData
IX	7949	2515	9158	0	1192	0	4.0075302	3.1900342	0	4.005533
KN	3760	1638	2792	0	1032	0	4.0225208	2.7632957	0	4.005533
KX	2756	1454	3522	0	856	0	4.0045314	3.0241339	0	4.0045342
NK	2540	1904	2312	0	744	0	4.0065279	2.8612366	0	4.0055325
NX	2756	1454	3522	0	856	0	4.0045314	3.0241339	0	4.0045342
XX	10124	1966	8460	2704	1129	0.015623	1.0033545	3.0140505	4.0137134	4.0026395

5.5.1 wM-Bus with NPF - Handshake Gateway

Gateway side presented in Table 5.10, which shows the data for six different handshake patterns in a wM-Bus system secured with the NPF, detailing both memory usage and time taken for each stage of the handshake process. The patterns analyzed are IX, KN, KX, NK, NX, and XX.

- Memory Usage: The XX pattern has significantly lower total memory usage compared to the other patterns, which might make it more suitable for devices with limited memory resources. The NX pattern shows the least memory usage among the other patterns, closely followed by NK, KX, KN, and IX in ascending order.
- Time Efficiency: The KN pattern appears to be the most time-efficient, with the shortest total time required for the handshake process. The NX, NK, and KX patterns show similar time efficiency, with the IX pattern being the least time-efficient among them. The XX pattern, despite its low memory usage, takes the longest total time, which might be due to the additional steps involved in its handshake process.

5.5.2 wM-Bus with NPF - Handshake Meter

Meter Side presented in Table 5.11, it presents data on six different handshake patterns in a wM-Bus system secured with the NPF, detailing both memory usage and elapsed time for the entire handshake process on the meter side.

- Memory Usage: The XX pattern has the highest total memory usage, potentially making it less suitable for devices with limited memory resources. The NK pattern

Table 5.12: Wireless M-Bus with Noise Framework Handshake between the Meter and Gateway

Pattern	Memory Init	Memory ClientHello	Memory Recv ServerHello	Memory ClientFinish	Memory AppData	Time Init	Time ClientHello	Time Recv ServerHello	Time ClientFinish	Time Recv AppData
IX	7949	155814	15616	0	2884	0	11.16312	7.2005641	0	7.1366014
KN	3760	147321	5834	0	2596	0	7.9271166	6.7688289	0	6.8887546
KX	2756	146133	6973	0	2276	0	7.9091246	7.0326619	0	7.1246116
NK	2540	146960	4906	0	2044	0	7.9920719	6.8657680	0	6.8877559
NX	2756	145740	6477	0	2044	0	7.9221160	7.0306649	0	7.1266117
XX	10124	16578	13527	6355	2776	0.015623	5.1693912	7.0322635	7.1821859	7.9594028

shows the least memory usage among the patterns, followed closely by KX, NX, and KN, with the IX pattern showing significantly higher memory usage.

- Time Efficiency: The KN pattern appears to be the most time-efficient, with the shortest total time required for the handshake process. The NK pattern is slightly less time-efficient, followed closely by the NX and KX patterns, which have identical time efficiencies. The IX pattern is less efficient than KN but more efficient than XX. The XX pattern, despite its high memory usage, does not have the highest time efficiency, potentially due to the additional steps involved in its handshake process.

5.5.3 wM-Bus with NPF - Handshake

Combining both sides Gateway and Meter handshake depicted in table 5.12, which detailing memory usage and elapsed time for the complete handshake process, including communication until data is sent in secure channels. The patterns analyzed are IX, KN, KX, NK, NX, and XX.

- Memory Usage vs. Time Efficiency: The XX pattern stands out with significantly lower total memory usage, making it an attractive option for systems with stringent memory constraints. However, this comes at the cost of the highest total time, indicating a trade-off between memory efficiency and time efficiency.
- High Memory Patterns: The IX pattern exhibits the highest memory usage, which may not be suitable for memory-constrained environments, despite its relatively high total time. The KN, KX, NX, and NK patterns have moderate memory usage and time requirements, presenting a balanced option for systems where both memory and time are considered.
- Optimal Choice: The choice between these patterns may depend on the specific requirements of the wM-Bus system. Systems that prioritize lower memory usage might lean towards the XX pattern, despite its longer time. In contrast, systems that require a balance between memory usage and time efficiency might prefer the KN,

KX, NX, or NK patterns.

Meter Memory Usage

Analyzing the Meter table into separate graphs as in figure 5.41, which present the memory usage of the initial step for the handshake from the Meter side. The category XX stands out with the highest memory usage at 10124 units. Category IX follows next, but it's significantly lower at 7949 units. The categories KN, KX, NK, and NX have similar memory usage, all below 4000 units, indicating that their memory requirements for handshake initiation are relatively lower compared to IX and XX. Figure 5.42. The bars for the 'IX', 'KN', 'KX', 'NK', and 'NX' patterns are relatively similar in height, suggesting that the memory usage for the ClientHello in these patterns is comparable and within the same range. However, the 'XX' pattern stands out due to its substantially lower memory usage. Figure 5.43. The chart displays a notable difference in memory usage between the patterns, with 'IX' and 'XX' using more memory in the ServerHello phase than the other patterns. The 'NK' pattern has the lowest memory usage, indicating it might be the most efficient or require fewer resources during this phase. This could be of interest when optimizing memory usage in systems where the ServerHello memory footprint is a consideration. Comparing this chart to the "ClientHello" chart, we see that the 'XX' category had notably less memory usage during the "ClientHello" but much higher usage during the "ServerHello," indicating a possible trade-off or difference in how each phase is handled within this pattern. This information is valuable for performance optimization and resource management during the handshake process of a secure communication protocol. Figure 5.44. presents the memory usage for transmitting app data from the meter to the gateway. The graph shows that the 'XX' category stands out with a much higher memory consumption than the others. This could indicate that the 'XX' category is either more resource-intensive, has memory leaks, or is performing more complex tasks requiring additional memory.

Meter Elapsed Time

In this section we will be Looking into the Elapsed Time of the Meter for each step of the handshake. In Figure 5.45. The 'IX' category has the longest elapsed time, indicating that the handshake process for this pattern may be the most complex or involves additional steps compared to others. The 'KN', 'KX', 'NK', and 'NX' categories have similar elapsed times, ranging from about 9.27 ms to 9.79 ms, suggesting that their processes are likely comparable in terms of time complexity. The 'XX' category stands out with the shortest elapsed time, which is significantly less than the others, implying that the 'XX'

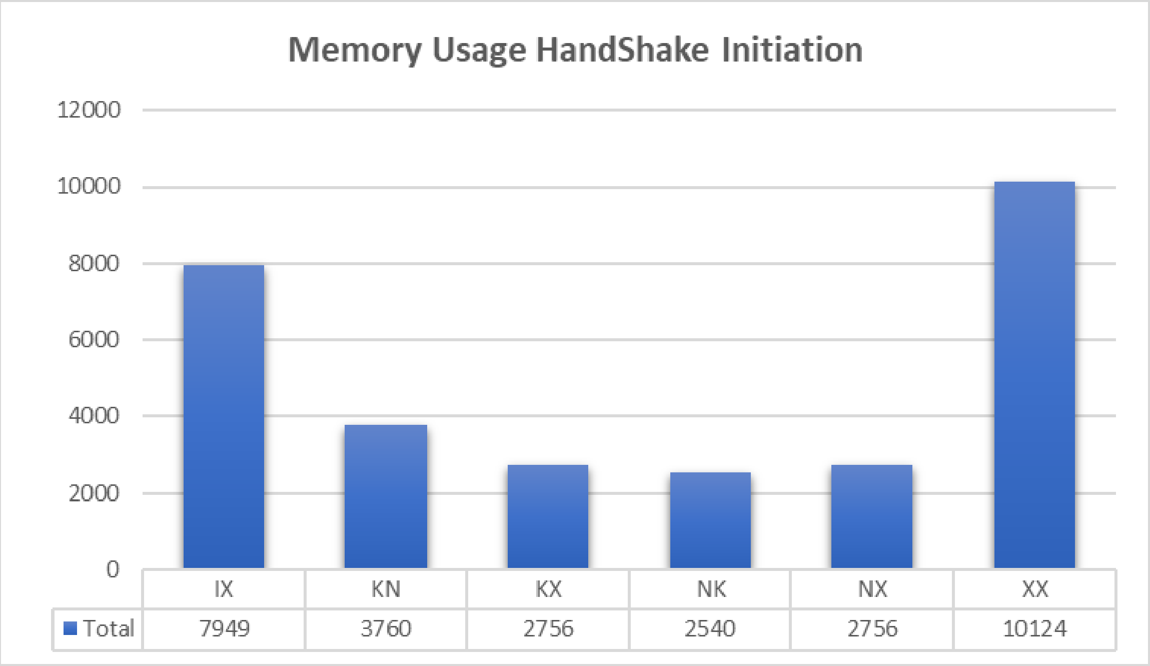


Figure 5.41: Memory Usage Handshake Initiate

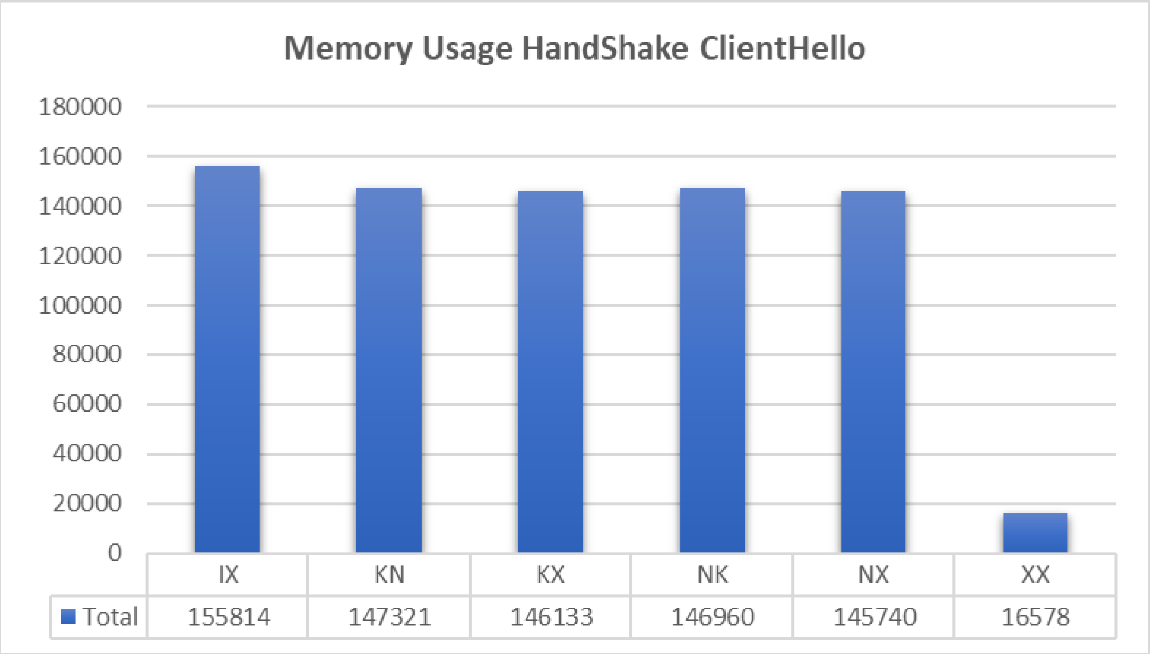


Figure 5.42: Memory Usage Handshake ClientHello

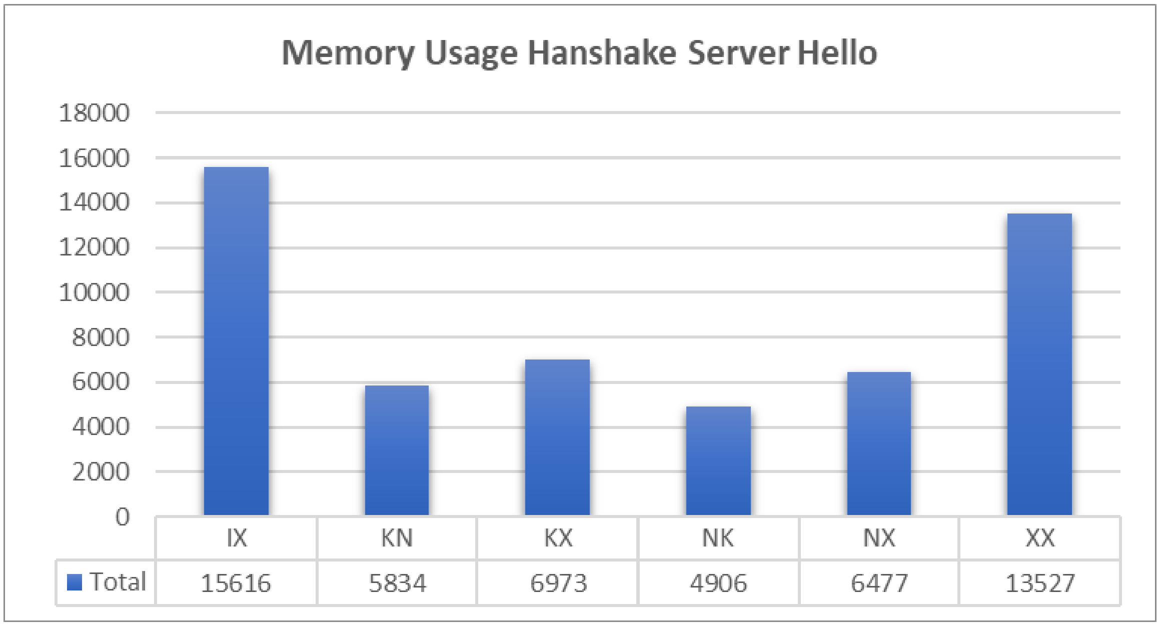


Figure 5.43: Memory Usage Handshake Receive ServerHello

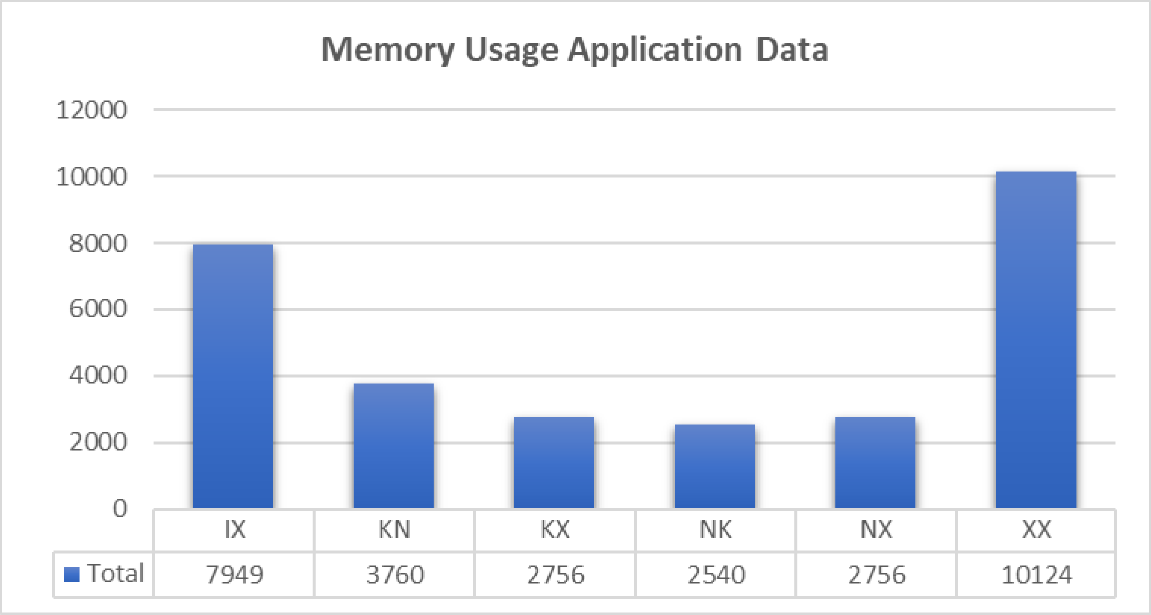


Figure 5.44: Memory Usage Handshake Send Application Data

pattern might be more efficient or simplified in terms of the ”ClientHello” phase. In addition the Figure 5.46 The ’IX’ pattern has the longest elapsed time for the ”ServerHello” phase among all the patterns. This is consistent with the ”ClientHello” phase, where the ’IX’ pattern also took the longest. The ’KN’ pattern shows the shortest elapsed time, which is different from the ”ClientHello” phase, where ’XX’ was the shortest. This suggests that the processes and operations involved in the ”ServerHello” phase for ’KN’ might be optimized for speed. The ’KX’, ’NK’, ’NX’, and ’XX’ categories have very similar elapsed times, all around 7 ms, indicating comparable efficiencies during the ”ServerHello” phase for these patterns. In Figure 5.47 Presents the elapsed time for the Meter sending ClientFinish, and its only shows in XX since the only pattern would have three messages the rest of the patterns runs under two messages. In Figure 5.48 The ’IX’, ’KX’, ’NK’, ’NX’ patterns have similar elapsed times, ranging from 7.12 ms to 7.14 ms, suggesting that the transmission time for application data is relatively consistent among these patterns. The ’KN’ and ’NK’ categories have the shortest elapsed times, both approximately 6.89 ms, indicating a faster transmission of application data which could be due to more efficient encoding, compression, or fewer data being transmitted compared to the other patterns. The ’XX’ pattern stands out with the longest elapsed time at 7.95 ms, which is notably higher than the other patterns. This could mean that the ’XX’ pattern involves additional processing, a larger amount of data, or less efficient data transmission methods for application data.

5.5.4 wM-Bus With NPF Handshake Cumulative Analysis

The cumulative memory usage for different phases of a handshake protocol, broken down by NPF patterns (IX, KN, KX, NK, NX, XX). presented in 5.49 we can analyze closely that:

- ’IX’: The memory usage is highest for all the phases combined. It is significantly higher for the ClientHello phase and the Application Data phase.
- ’KN’: The memory usage is much lower across all phases compared to ’IX’, with ClientFinish and Application Data being the most substantial contributors.
- ’KX’: Shows a similar pattern to ’IX’ with a slightly lower total memory usage. ClientHello and Application Data phases dominate the memory usage.
- ’NK’: Comparable to ’KX’, with a major part of memory used during the ClientHello and Application Data phases.

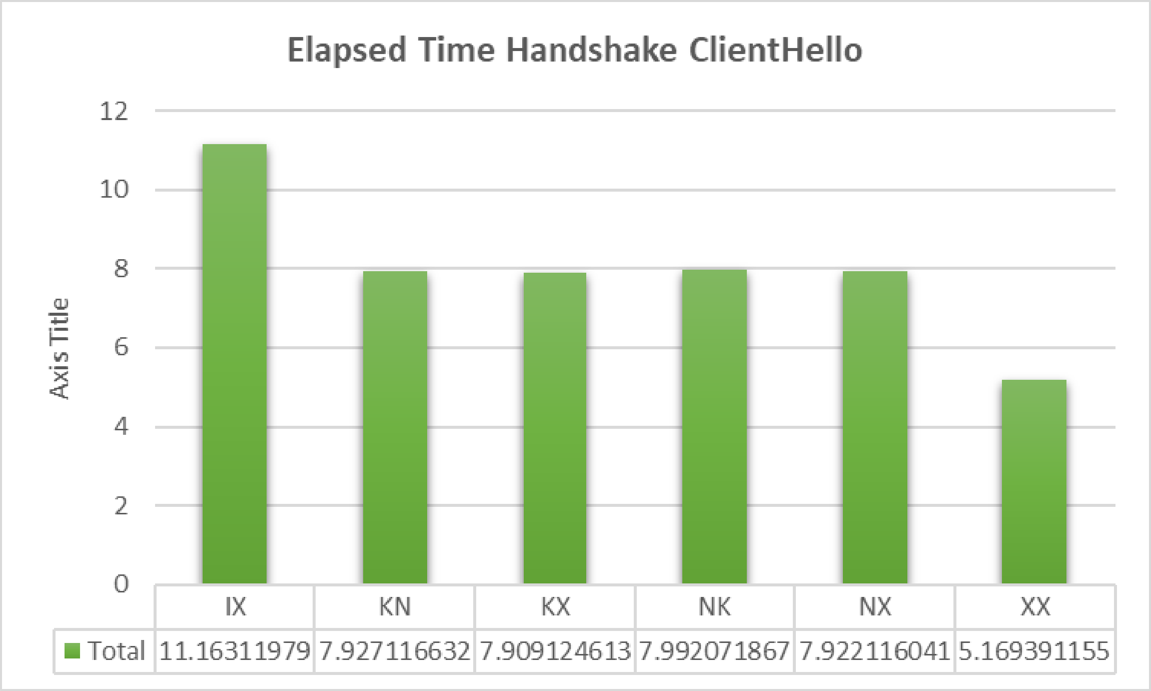


Figure 5.45: Memory Usage Handshake Initiate

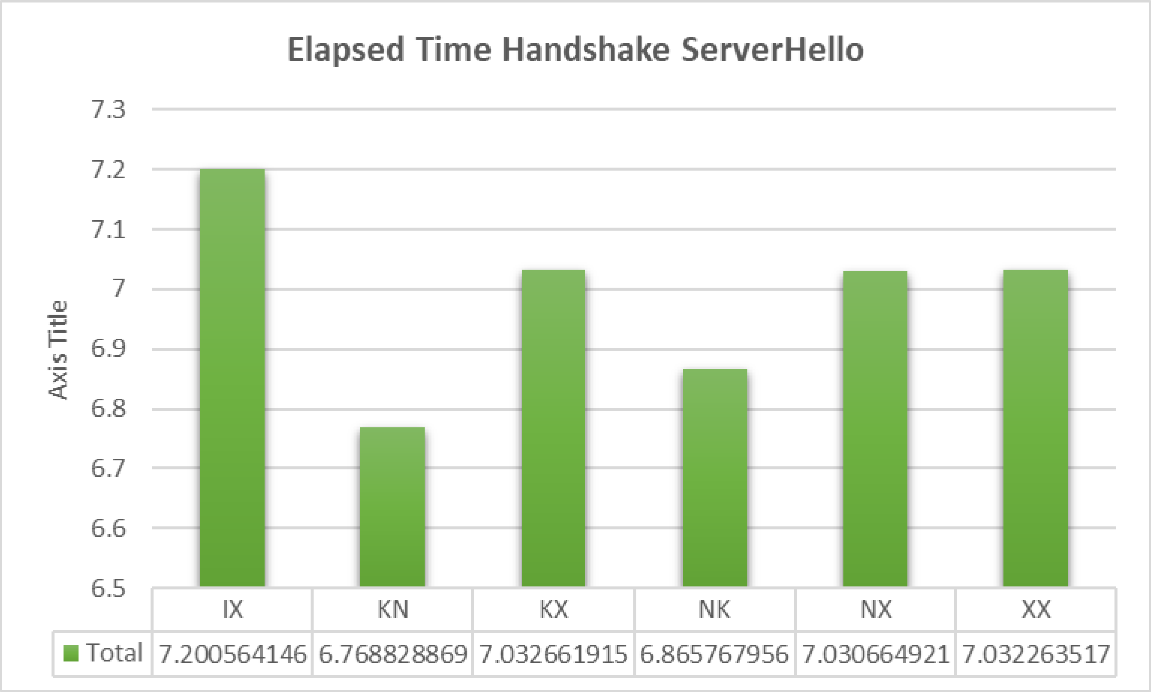


Figure 5.46: Memory Usage Handshake ClientHello

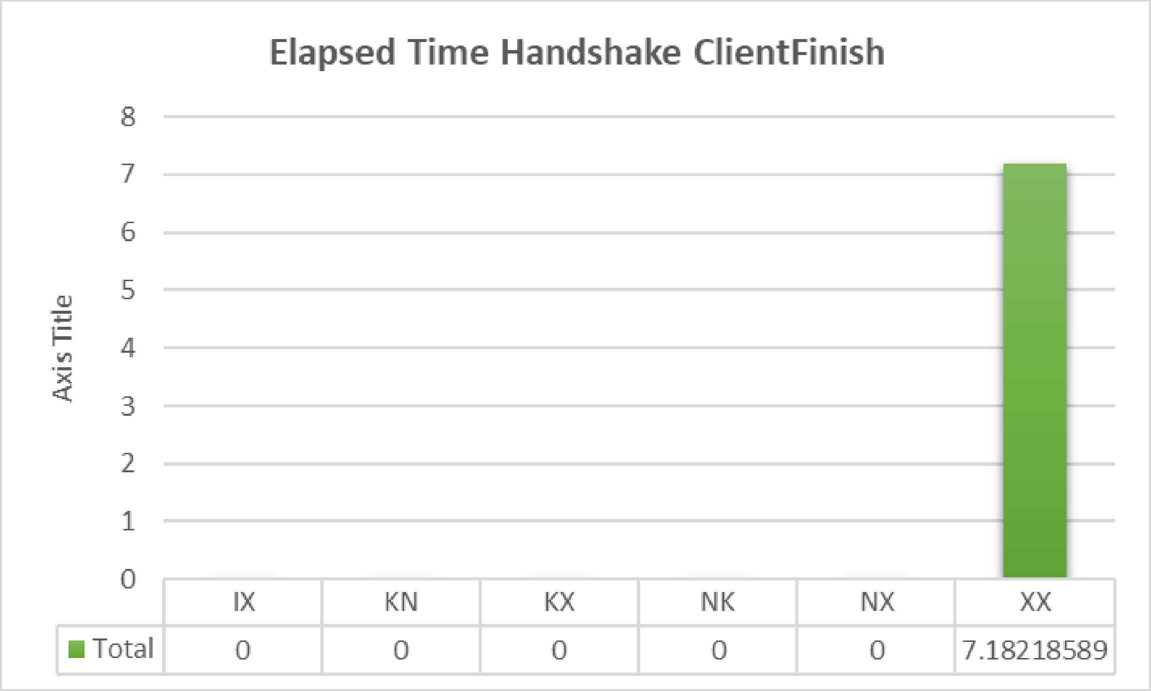


Figure 5.47: Memory Usage Handshake Receive ServerHello

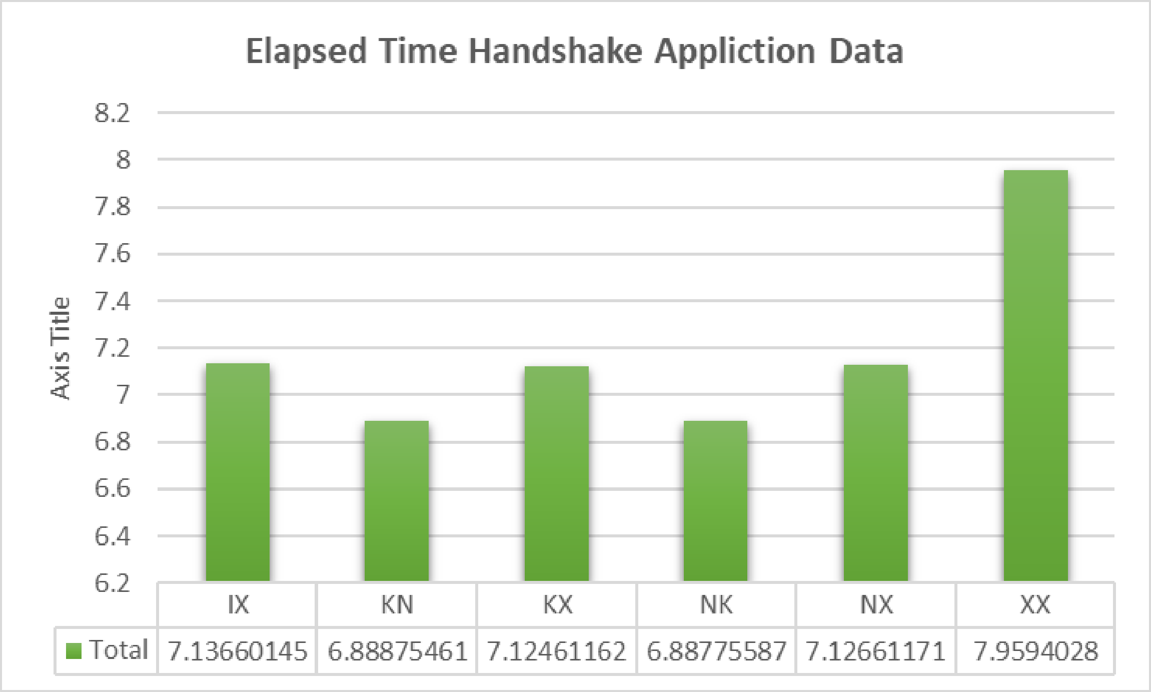


Figure 5.48: Memory Usage Handshake Send Application Data

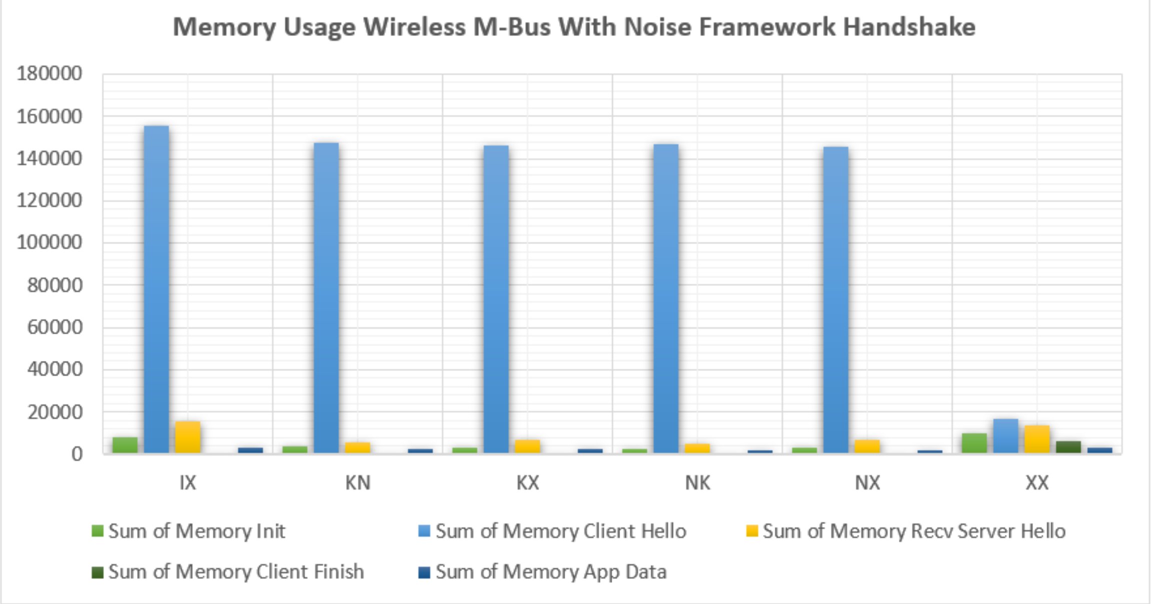


Figure 5.49: Memory Usage wM-Bus With NPF Handshake for all Handshake steps

- ’NX’: Similar to ’NK’ and ’KX’ in the distribution of memory usage among the phases.
- ’XX’: Stands out with a significantly lower overall memory usage. The ClientFinish phase appears to be the largest contributor, while the other phases are relatively minimal.

It can be inferred that ’IX’ is the most memory-intensive pattern overall, while ’XX’ is the least. This data can be crucial for performance optimization and resource management during the handshake process in secure communication protocols, especially when working with resource-constrained devices such as those using wM-Bus.

Moreover, the cumulative Elapsed Time for the handshake phases is shown in Figure 5.50. Here’s an analysis:

- ’IX’ category: Has the highest initial time (Init) and notable times for ClientHello and receiving Application Data. The ClientFinish and ServerHello times are less prominent.
- ’KN’ category: Shows a more balanced distribution of time across all phases, with the ServerHello and receiving Application Data phases occupying the most time.
- ’KX’ category: Similar to ’KN’, but with a slightly longer time for ClientFinish and

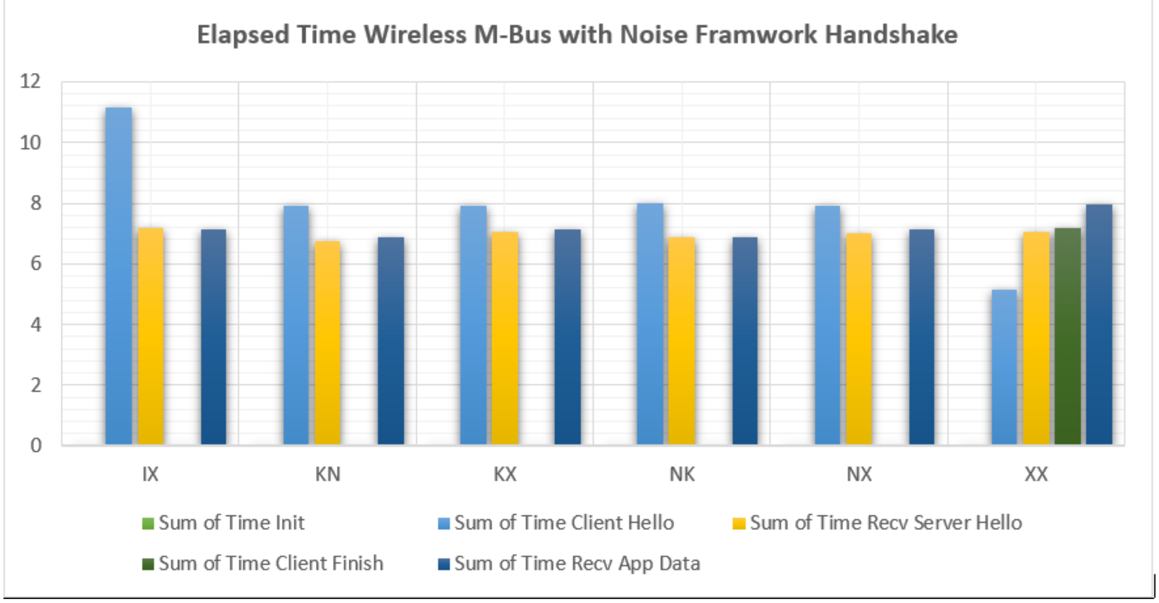


Figure 5.50: Elapsed Time wM-Bus with NPF Handshake for all Handshake steps

a shorter time for ServerHello.

- ’NK’ category: The time is more evenly distributed across all phases, with none being extremely dominant, but with ClientFinish and receiving Application Data being slightly higher.
- ’NX’ category: Shows a similar pattern to ’NK’, with a notable portion of time spent on ClientFinish and receiving Application Data.
- ’XX’ category: This pattern stands out with a very different distribution. The time for receiving Application Data is the most significant, followed by ServerHello, with relatively smaller amounts of time for Init, ClientHello, and ClientFinish.

This chart illustrates the comparative efficiency of each NPF pattern in terms of time consumption for each phase of the handshake process. ’IX’ has a high initial setup time, while ’XX’ is distinct in that it spends most of its time in the later phases (receiving Application Data).

The final view for this phase as depicted in Figure 5.51 the total memory usage of the Entire handshake and sending application data. Analyzing each pattern:

- ’IX’: Shows the highest total memory usage among the patterns, with a usage of 182,263 units.

- ’KN’: Displays a total memory usage of 159,511 units, which is lower than the ’IX’ pattern but still on the higher side.
- ’KX’: The memory usage is slightly less than ’KN’, standing at 158,138 units.
- ’NK’: Shows a total memory usage of 156,450 units, which is close to that of ’KX’.
- ’NX’: Has a total memory usage of 157,017 units, comparable to ’NK’ and ’KX’.
- ’XX’: Stands out with significantly lower total memory usage, at 49,360 units, which is less than a third of the usage compared to the other patterns.

The chart indicates that the ’XX’ NPF pattern is the most memory-efficient among those listed. In contrast, ’IX’ is the most memory intensive. This information is particularly useful when memory resources are a critical constraint and could influence the selection of a NPF pattern for implementation in memory-limited environments. As for the Elapsed Time for the handshake as depicted in Figure 5.52, which presents the six patterns of NPF. Here’s a detailed analysis of each pattern:

- ’IX’: This has the shortest total elapsed time at approximately 25.50 sec, indicating a relatively quick handshake process.
- ’KN’: Shows a total elapsed time of around 21.58 sec, which is less than ’IX’, suggesting a more efficient handshake in terms of time.
- ’KX’: The elapsed time is about 22.06 sec, similar to ’KN’ and also indicating efficiency.
- ’NK’: Has a total elapsed time of approximately 21.75 sec, very close to ’KN’ and ’KX’.
- ’NX’: Shows a slightly increased total elapsed time of around 22.07 sec.
- ’XX’: Has the longest total elapsed time at approximately 27.36 sec, which is noticeably higher than the other patterns.

The chart indicates that the ’KN’, ’KX’, and ’NK’ patterns are the most time-efficient for the handshake process, with very similar total elapsed times. The ’IX’ pattern, despite having a higher total time than ’KN’, ’KX’, and ’NK’, is still quite efficient. However, the ’XX’ pattern stands out as having the longest total elapsed time, suggesting it may involve more complex or time-consuming operations during the handshake process. This information can be vital for optimizing the performance of wM-Bus systems, especially when the handshake duration is critical.

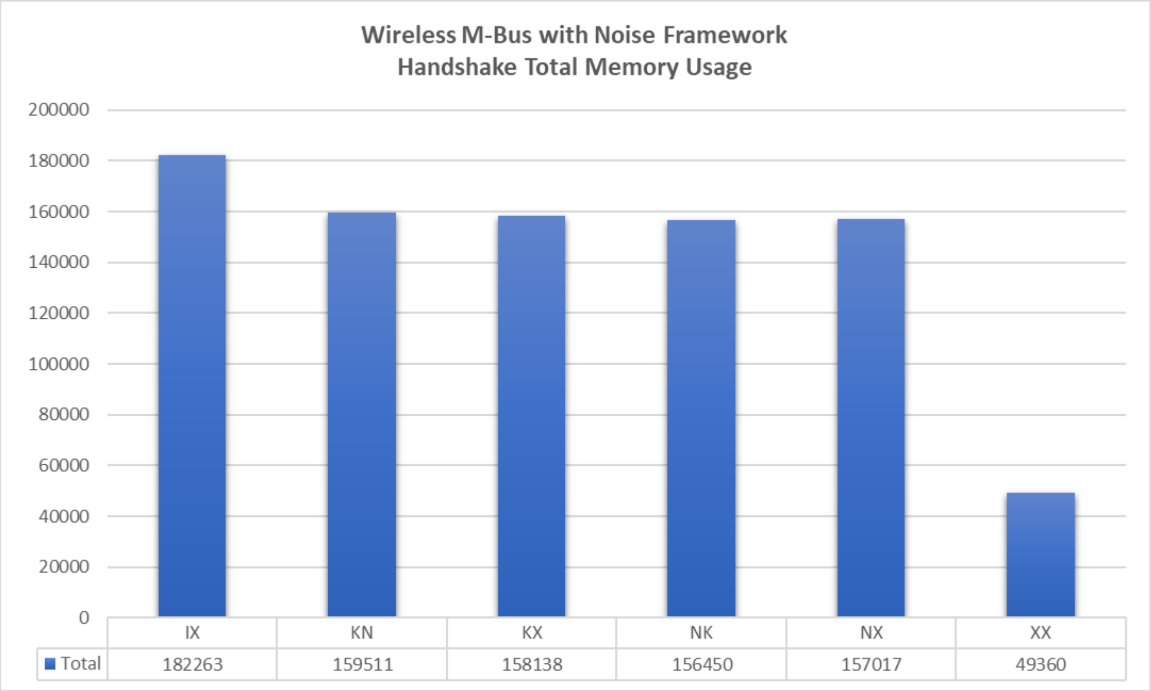


Figure 5.51: wM-Bus with NPF Handshake Total Memory Usage

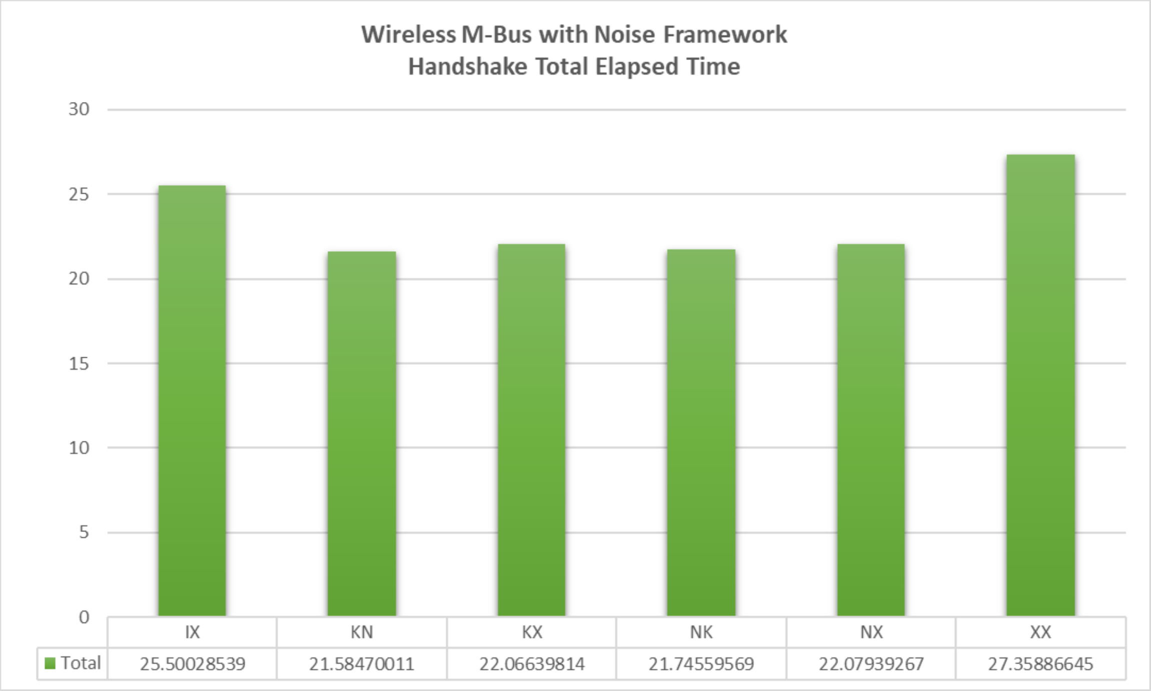


Figure 5.52: wM-Bus with NPF Handshake Total Elapsed Time

5.6 Phase Five: Optimizing NPF for Lightweight Protocol

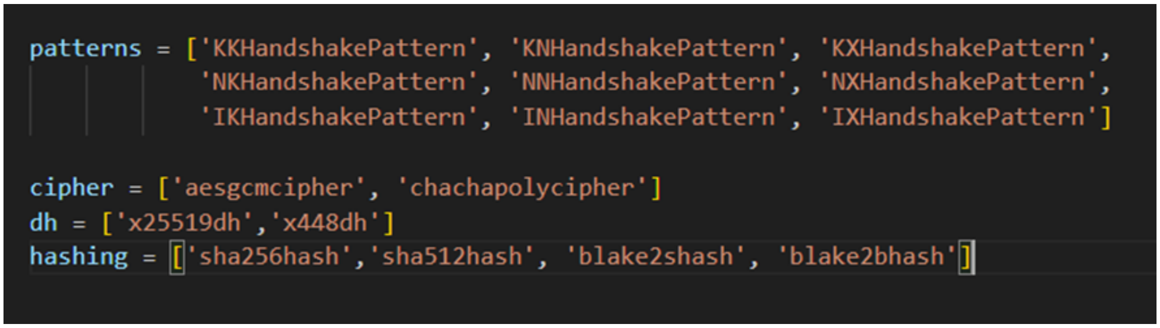
In this phase of our investigation, we consistently employed the identical hardware setup for the Radiocrafts cards, specifically the ”RC1701HP-WMBUS4” model, to accurately replicate the functionality of both the meter and the gateway within our experimental framework. To enhance the robustness of the NPF, we did not limit ourselves to merely executing pre-established patterns. Instead, we adopted a more dynamic approach by integrating comprehensive security properties into the Python code we developed, thereby optimizing the overall system performance, and ensuring a higher level of security and reliability in data transmission.

As well, we extended our methodology by incorporating specialized Python libraries tailored for wM-Bus communication, alongside the utilization of the NPF to bolster our security protocols. We also integrated the tracemalloc module to meticulously monitor memory consumption, establishing it as a crucial metric alongside others such as the time duration required to complete the entire handshake process and the size of the transmitted packets.

We delved into specific cryptographic primitives, selecting AES-GCM and ChaCha20-Poly1305 as our cipher suites for their robust security and efficiency. For the Diffie-Hellman key exchange, we opted for the X25519 and X448 curves, known for their speed and security. In terms of hashing algorithms, our focus was on SHA-256, SHA-512, Blake2s, and Blake2b, each chosen for their cryptographic strength and performance.

Our procedural approach mirrored the established practices in TLS, adapting them within the context of the NPF. This entailed initiating the handshake, followed by the exchange of ’client hello’ and ’server hello’ messages. Depending on the protocol variant, this could culminate in a ’client finish’ message in a three-message exchange or proceed without it in a two-message scenario. The subsequent encryption and decryption of communications are contingent upon the pre-agreed combination of cipher suite, Diffie-Hellman mechanism, and hashing algorithm, ensuring a secure and efficient cryptographic handshake. During the experiment, we meticulously executed 12 distinct patterns as depicted in Figure 5.53, each paired with approximately 16 security configurations to thoroughly evaluate their performance and compatibility.

The patterns are:



```
patterns = ['KKHandshakePattern', 'KNHandshakePattern', 'KXHandshakePattern',
            'NKHandshakePattern', 'NNHandshakePattern', 'NXHandshakePattern',
            'IKHandshakePattern', 'INHandshakePattern', 'IXHandshakePattern']

cipher = ['aesgcmcipher', 'chachapolycipher']
dh = ['x25519dh','x448dh']
hashing = ['sha256hash','sha512hash', 'blake2shash', 'blake2bhash']
```

Figure 5.53: Sample of the iteration of all patterns with various ciphers, key exchange (DH), and hashing algorithms

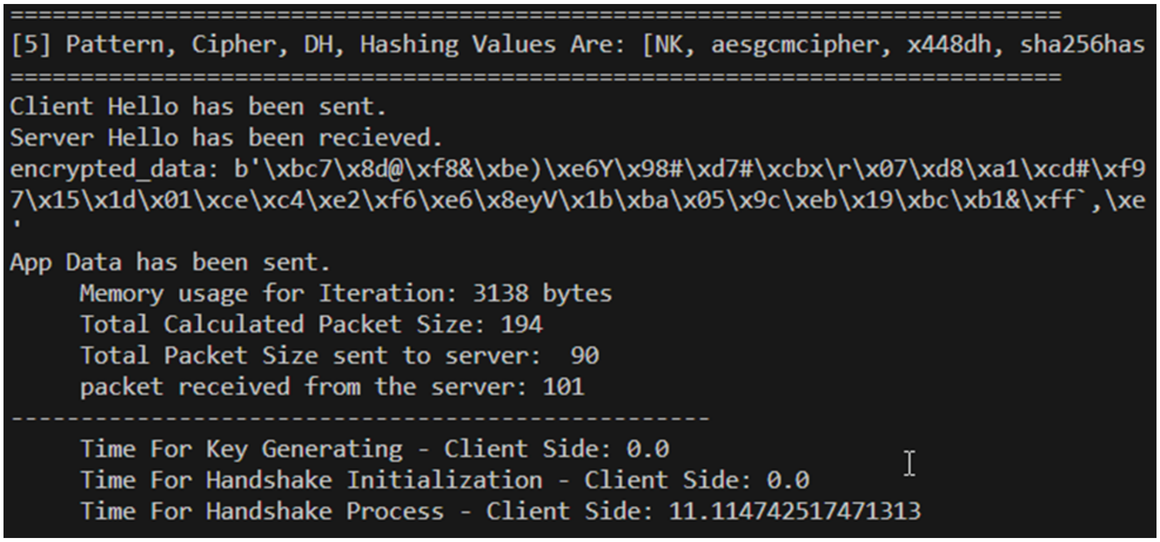
- Two Messages: IK, IN, IX, KK, KN, KX, NK, NN, NX
- Three Message: XK, XN, XX

However, two patterns with specific security options were deliberately omitted from our trials. The rationale behind this exclusion was the oversized packet length these patterns generated, surpassing the maximum allowable dimensions defined by the wM-Bus protocol. This limitation has been earmarked for future investigation to potentially adapt and integrate these patterns within our framework.

- Two Messages: IK, IN, IX, KX, NX with DH option X448
- Three Messages: XX with DH option X448

Each execution cycle of our experiment was methodically recorded, with outcomes being systematically documented in a dedicated log file. A sample terminal view for the execution depicted in Figure 5.54. This approach ensured a detailed and accessible record of the results, facilitating an in-depth analysis and comparison of the various security configurations tested. The accumulated data, alongside illustrative figures, provide a comprehensive overview of the experiment's findings, highlighting the efficiency, reliability, and security implications of each tested pattern and configuration.

In the forthcoming section, we will systematically present the outcomes associated with each identified pattern through comprehensive tables. Subsequent to each table, a corresponding graphical representation will be provided to visually illustrate the data. This will be followed by an in-depth analysis of the results, aimed at elucidating the implications and insights derived from the observed patterns.



```
===========================================================================
[5] Pattern, Cipher, DH, Hashing Values Are: [NK, aesgcmcipher, x448dh, sha256has
===========================================================================
Client Hello has been sent.
Server Hello has been recieved.
encrypted_data: b'\xbc7\x8d@\xf8&\xbe)\xe6Y\x98#\xd7#\xcbx\r\x07\xd8\xa1\xcd#\xf9
7\x15\x1d\x01\xce\xc4\xe2\xf6\xe6\x8eyV\x1b\xba\x05\x9c\xeb\x19\xbc\xb1&\xff`,\xe
'
App Data has been sent.
     Memory usage for Iteration: 3138 bytes
     Total Calculated Packet Size: 194
     Total Packet Size sent to server:  90
     packet received from the server: 101
--------------------------------------------------
     Time For Key Generating - Client Side: 0.0
     Time For Handshake Initialization - Client Side: 0.0
     Time For Handshake Process - Client Side: 11.114742517471313
```

Figure 5.54: Sample of the terminal view during the execution of the patterns with different security options

5.6.1 NPF - Two Messages

IK Pattern

Particularly, we focused on the ”IK” pattern, which was thoroughly explored. This pattern’s intricate details were illustrated in Figures 5.55 through 5.63, providing a comprehensive view of the experimental findings related to this specific pattern. Furthermore, a summary of the overall data was encapsulated in Figures 5.61, 5.62, and 5.63, which aggregated the total values derived from each of the graphs previously mentioned. For the remaining patterns, only the cumulative data presented in the last three graphs was discussed, offering a concise overview of these patterns without delving into the same level of detail as provided for the ”IK” pattern. This structured approach enabled a clear and systematic comparison of the different patterns, highlighting the unique characteristics and implications of each, particularly the ”IK” pattern, within the context of network performance and efficiency.

1. **IK handshake memory usage** depicted in Figure 5.55 we can observe the following:
    - All configurations have relatively similar memory usage, with values around the 8000 to 12000 bytes range. This suggests that the memory footprint of each configuration is comparable.
    - There doesn’t appear to be a significant difference in memory usage between the

  use of AES-GCM and ChaCha20 encryption methods, nor between the different hashing algorithms (Blake2b, Blake2s, SHA-256, SHA-512).
   - The memory usage is consistently below 12000 bytes for all options, indicating that all configurations are relatively lightweight in terms of memory consumption.

2. **IK handshake packet size** depicted in Figure 5.56, we can observe the following:
   - The ”aesgcm-25519-blake2b” option has the largest packet size, exceeding 780 bytes.
   - The ”aesgcm-25519-blake2s” and ”aesgcm-25519-sha512” configurations result in smaller packet sizes, around 680 bytes.
   - The ”aesgcm-25519-sha256” configuration has a packet size just slightly above 700 bytes.
   - The ”chacha20-25519-blake2b” option has a packet size around the mid-700 bytes range.
   - The ”chacha20-25519-blake2s” and ”chacha20-25519-sha256” configurations have similar packet sizes, both around 660 bytes.
   - The ”chacha20-25519-sha512” configuration results in the second-largest packet size, close to 800 bytes.

3. **IK handshake elapsed time** depicted in Figure 5.57 we can observe the following:
   - The fastest handshake times are with configurations ”aesgcm-25519-blake2b” and ”aesgcm-25519-sha256,” which indicates that the AES-GCM encryption with either Blake2b or SHA-256 hashing algorithms in conjunction with Curve25519 for the Diffie-Hellman function offers the quickest handshake times in this set of data.
   - The ”chacha20-25519-blake2b” configuration shows a noticeable increase in time compared to the AES-GCM options, suggesting that in this instance, ChaCha20 may be slower for handshakes than AES-GCM.
   - The ”chacha20-25519-blake2s” and ”chacha20-25519-sha256” configurations also have longer handshake times but are faster than the ”chacha20-25519-blake2b” option.

Table 5.13: IK Pattern Handshake - Meter (Client)

No	**Handshake Options**	**Memory Usage Bytes**	**Pocket Size ClientHello**	**Packet Size ServerHello**	**Packet size AppData**	**Packet Size Sent Bytes**	**Packet Size Recv Bytes**	**Time KeyGen Sec**	**Time Handshake Initiate Sec**	**Time Handshake Process Sec**
1	aesgcm-x2551-sha256	3020	194	146	152	114	77	0.009124	0.001	11.10603
2	aesgcm-x2551-sha512	3148	226	178	184	114	77	0	0	11.11609
3	aesgcm-x2551-blake2s	3021	194	146	152	114	77	0	0	11.11548
4	aesgcm-x2551-blake2b	3149	226	178	184	114	77	0	0	11.11569
5	chacha-x2551-sha256	4073	190	142	148	114	77	0	0.006727	11.10943
6	chacha-x2551-sha512	4201	222	174	180	114	77	0	0	11.11557
7	chacha-x2551-blake2s	4074	190	142	148	114	77	0	0	11.11617
8	chacha-x2551-blake2b	4202	222	174	180	114	77	0	0	11.10205

Table 5.14: IK Pattern Handshake - Gateway (Server)

No	**Handshake Options**	**Memory Usage Bytes**	**Packet size ClientHello**	**Packet Size ServerHello**	**Packet Size AppData**	**Packet Size Sent Byte**	**Packet size Recv Bytes**	**Time Handshake Initiate Sec**	**Time Handshake Process Sec**
1	aesgcm-x2551-sha256	5550	194	146	152	66	125	0	6.899738
2	aesgcm-x2551-sha512	5670	226	178	184	66	125	0	6.904554
3	aesgcm-x2551-blake2s	5536	194	146	152	66	125	0	6.904407
4	aesgcm-x2551-blake2b	5656	226	178	184	66	125	0	6.904418
5	chacha-x2551-sha256	6907	190	142	148	66	125	0	6.886844
6	chacha-x2551-sha512	6960	222	174	180	66	125	0	6.888678
7	chacha-x2551-blake2s	6893	190	142	148	66	125	0	6.904415
8	chacha-x2551-blake2b	7005	222	174	180	66	125	0	6.890923

- There is a significant increase in handshake time for the ”chacha20-25519-sha512” option, indicating that using SHA-512 hashing significantly increases the handshake duration compared to the other options.

IN Pattern

The ”IN” pattern within the Noise Framework was closely examined alongside twelve other patterns, focusing on its unique handshake communication. This analysis was clearly presented through two tables 5.15 and 5.16, detailing the process from the perspectives of both the meter and the gateway, and was followed by a chart that summarized the findings. This chart provided a clear overview of memory usage, packet size, and the time required to complete the handshake and transmit meter data.

1. **IN handshake memory usage** depicted in Figure 5.64 we can observe the following:
    - The aesgcm-25519-blake2b option uses the most memory, closely followed by the chacha20-25519-sha512 option. Both are significantly higher in memory usage compared to the other options.
    - The aesgcm-25519-sha256 uses the least memory.
    - Memory usage for the chacha20-25519-blake2s and chacha20-25519-sha512

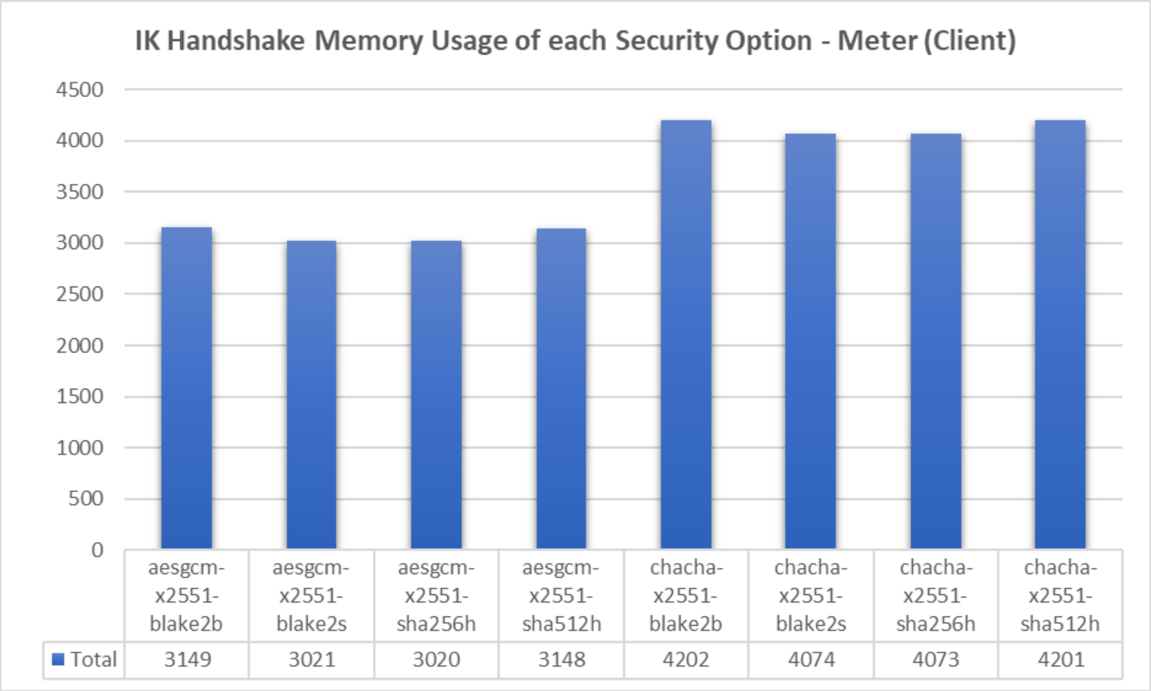


Figure 5.55: Meter Memory Usage associated with different security options in the IK pattern

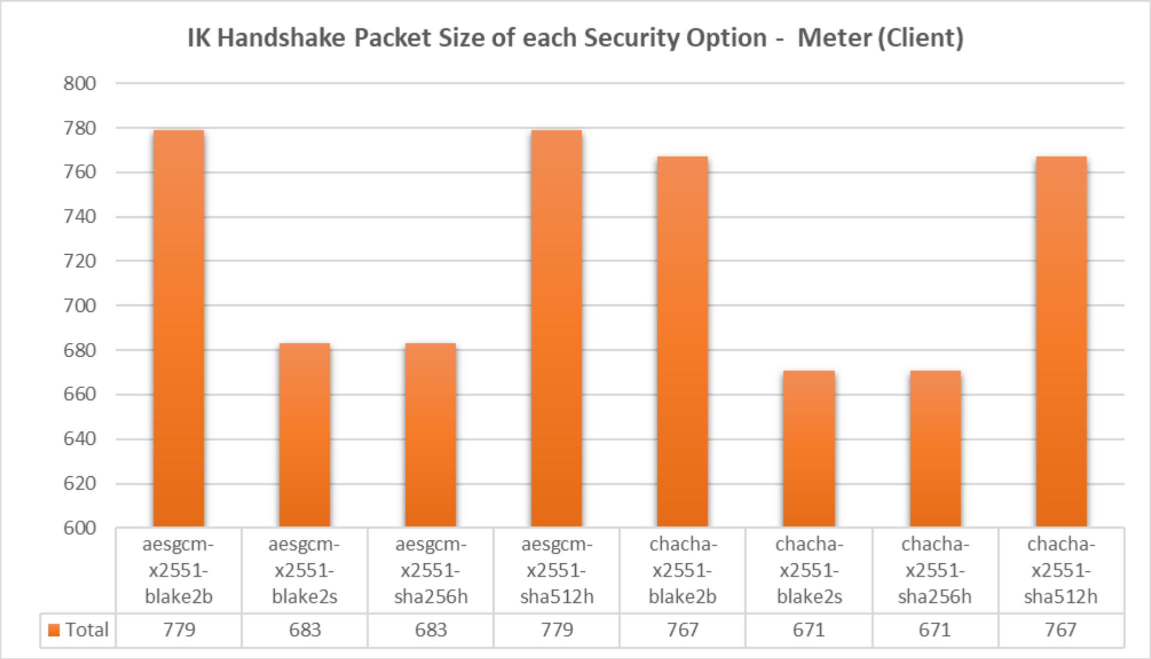


Figure 5.56: Meter Packet size associated with different security options in the IK pattern

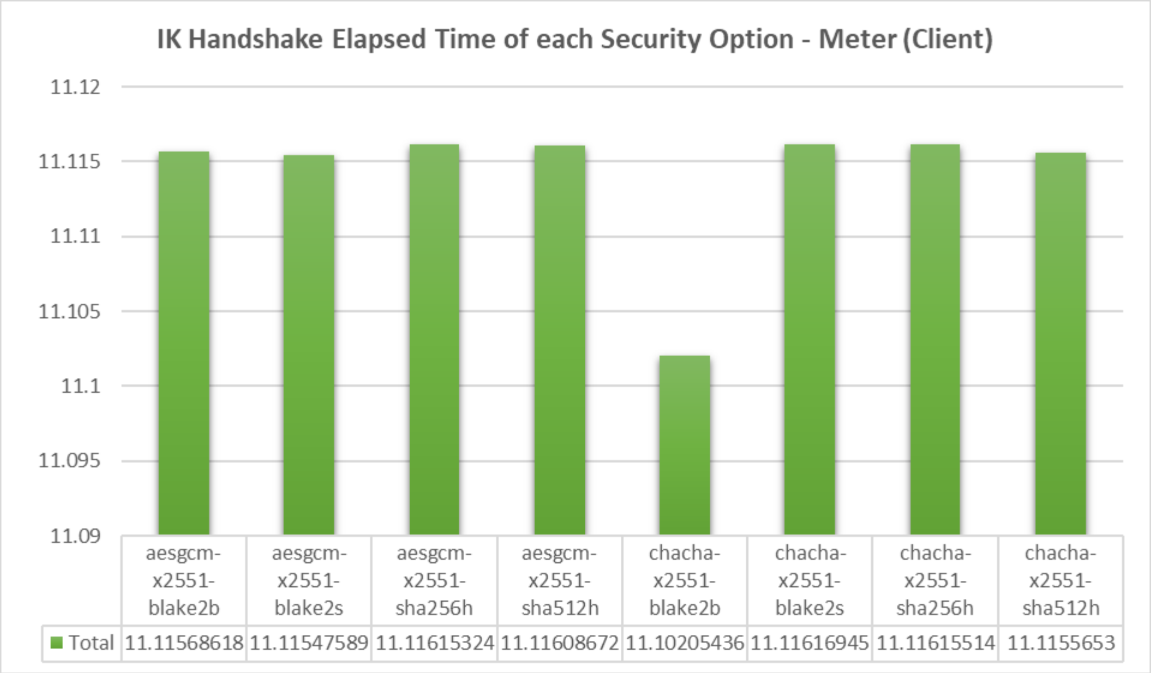


Figure 5.57: Meter Elapsed Time associated with different security options in the IK pattern

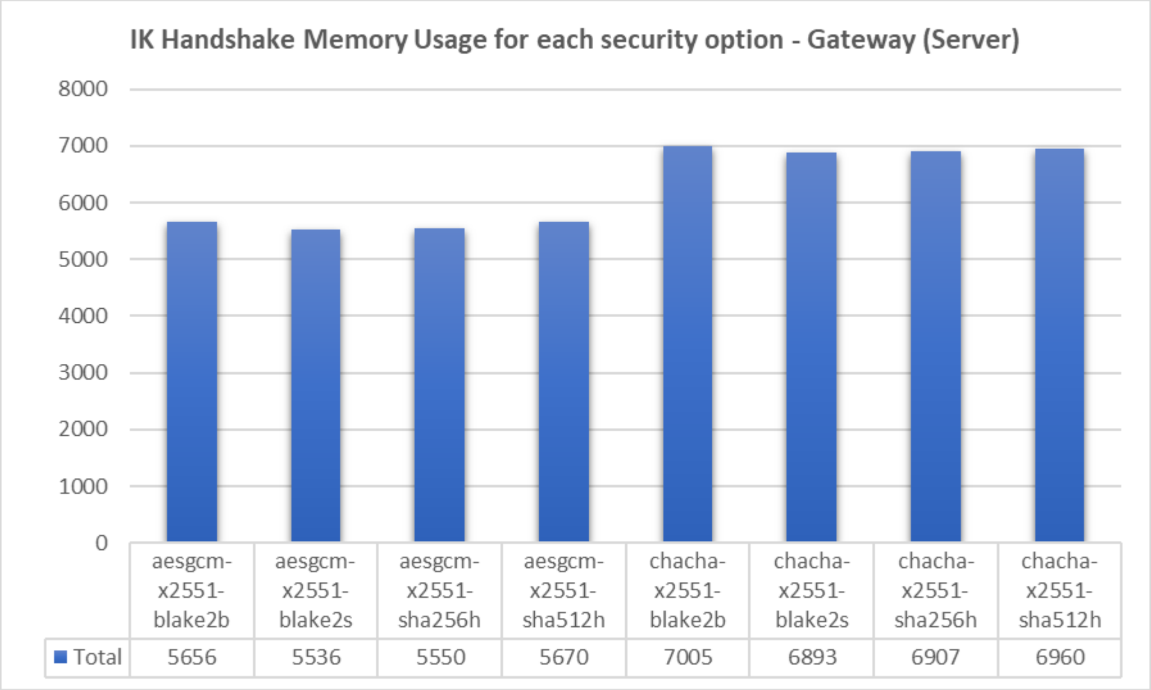


Figure 5.58: Gateway Memory Usage associated with different security options in the IK pattern

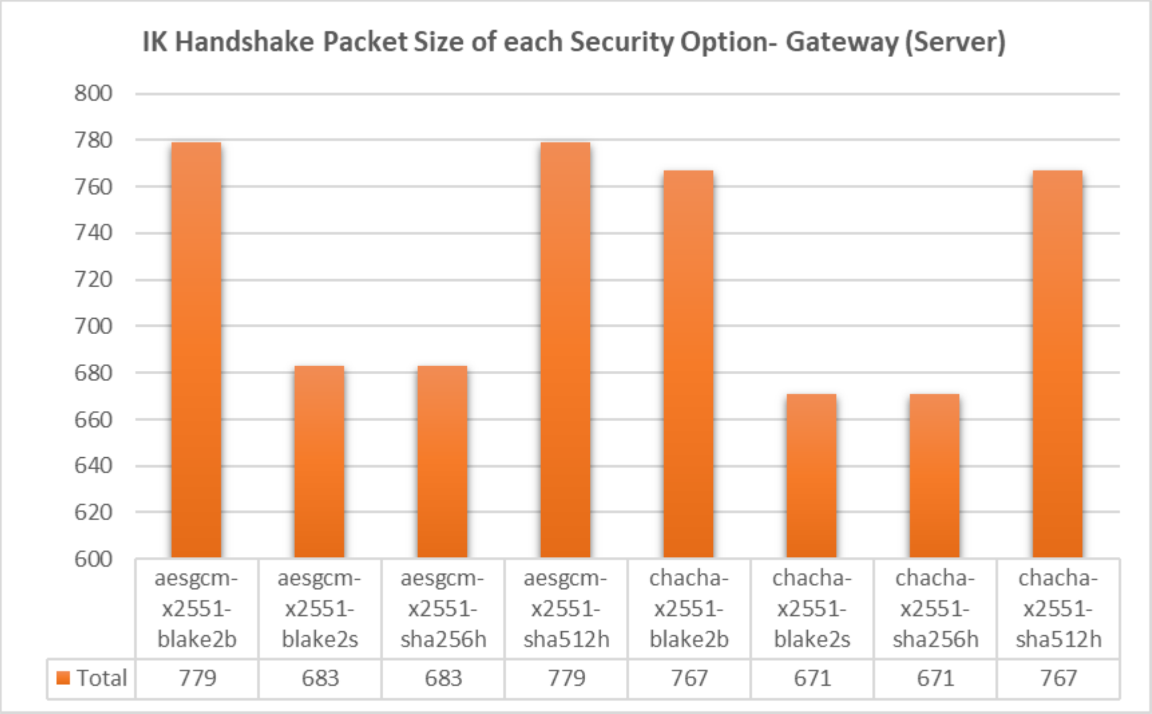


Figure 5.59: Gateway Packet Size associated with different security options in the IK pattern

options is very similar and on the higher end of the spectrum.

- The options aesgcm-25519-blake2s and aesgcm-25519-sha512 fall in the middle range of memory usage, with aesgcm-25519-sha512 being slightly higher.

2. **IN handshake packet size** depicted in Figure 5.65, we can observe the following:
   - The largest packet sizes are observed when using aesgcm-x2551-blake2b and aesgcm-x2551-sha512 configurations, both exceeding 1400 bytes.
   - The smallest packet sizes are observed with the chacha-x2551-blake2s and chacha-x2551-sha256, both around 1214 bytes, indicating a more compact packet size in these configurations.
   - The packet size for aesgcm-x2551-blake2s and aesgcm-x2551-sha256 configurations are identical, both at approximately 1238 bytes.
   - The chacha-x2551-blake2b and chacha-x2551-sha512 configurations have packet sizes that are notably larger than their blake2s and sha256 counterparts but are still smaller than the largest aesgcm configurations.

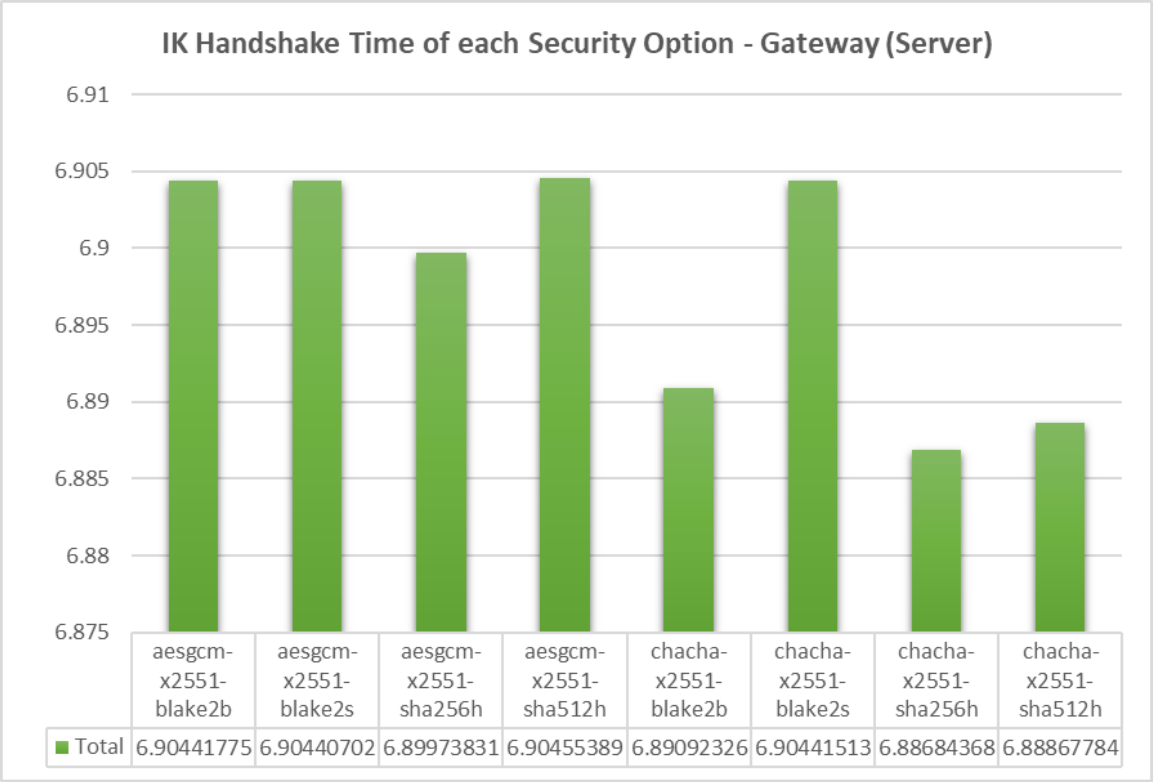


Figure 5.60: Gateway Elapsed Time associated with different security options in the IK pattern

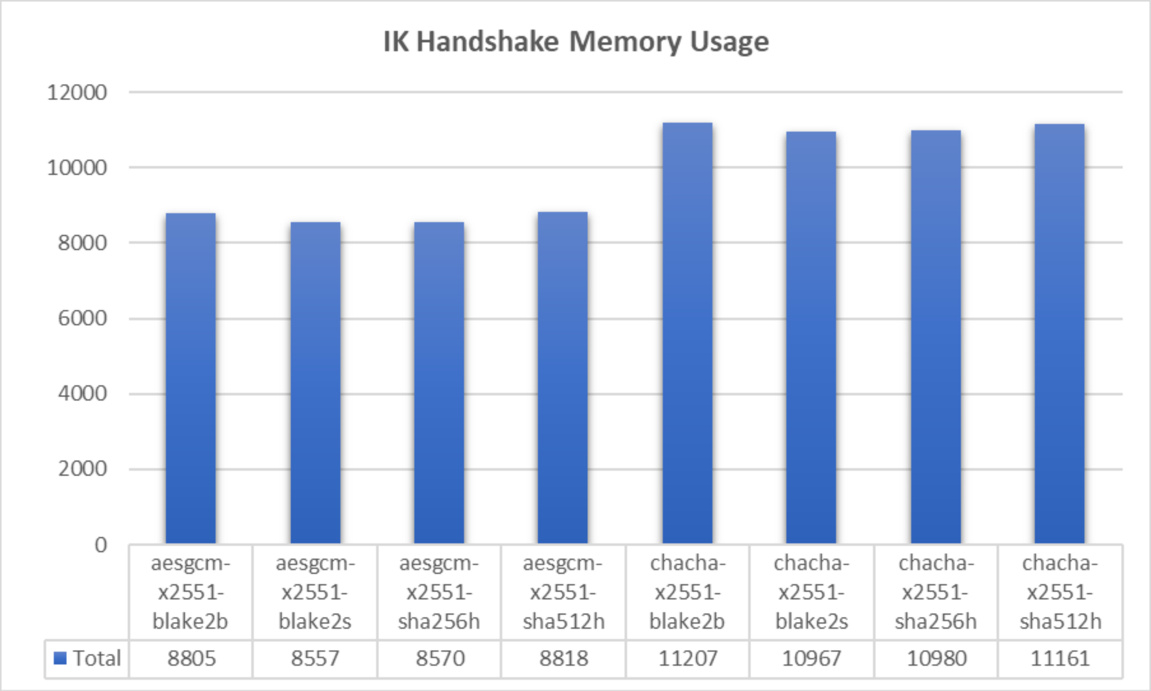


Figure 5.61: IK Handshake Memory Usage associated with different security options

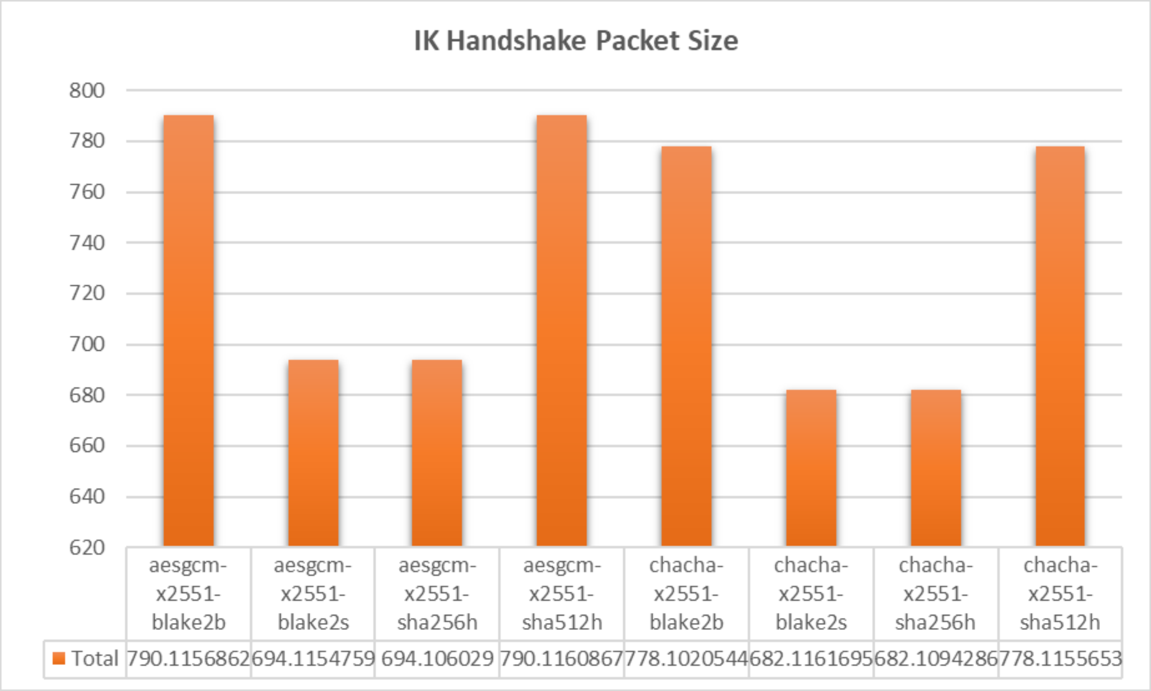


Figure 5.62: IK Handshake Packet Size associated with different security options

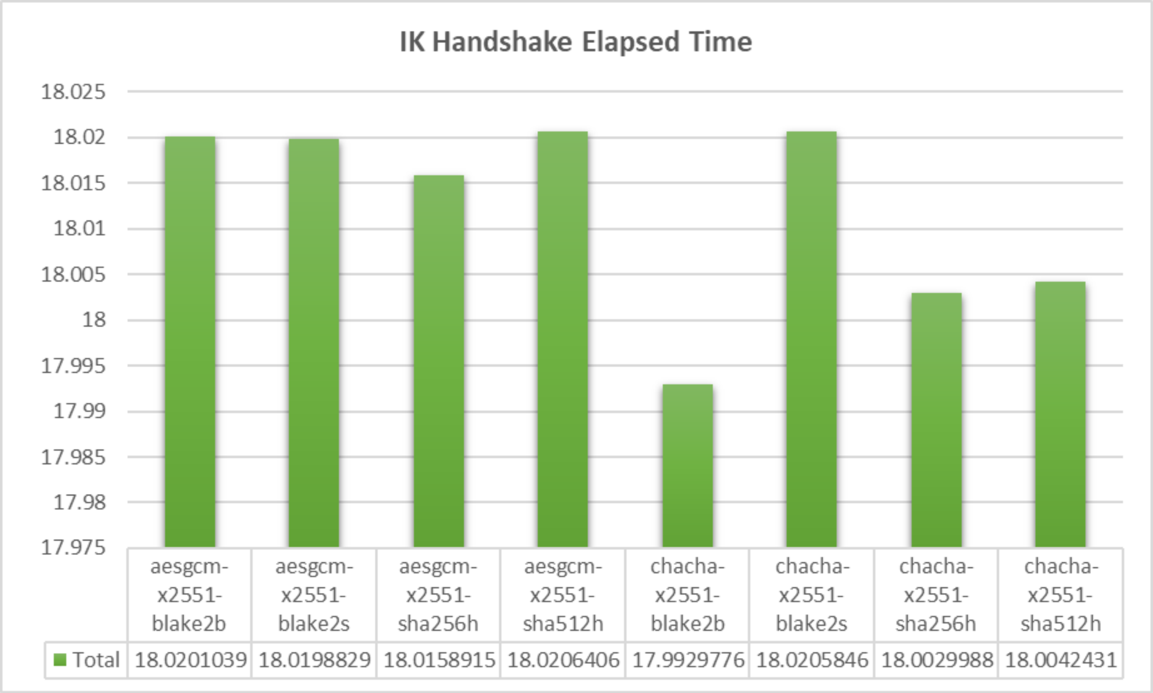


Figure 5.63: IK Handshake Elapsed Time associated with different security options

- The choice of Cipher (aesgcm vs chacha) appears to have a more significant impact on packet size than the choice of Hash (blake2b, blake2s, sha256, sha512).
- Overall, the chacha Cipher options consistently result in smaller packet sizes than the aesgcm options for the corresponding Hash functions.

3. **IN handshake elapsed time** depicted in Figure 5.66 we can observe the following:
    - The aesgcm-25519-blake2b and chacha20-25519-sha512 options have the longest handshake times, indicating that they may be the most time-intensive processes.
    - The aesgcm-25519-sha256 has the shortest handshake time, suggesting it is the most time-efficient option among those listed.
    - The other options (aesgcm-25519-blake2s, aesgcm-25519-sha512, chacha20-25519-blake2b, chacha20-25519-blake2s) have handshake times that are between the longest and shortest times, with aesgcm-25519-sha512 being faster than chacha20-25519-blake2b but slower than aesgcm-25519-sha256.

Table 5.15: IN Pattern Handshake - Meter (Client)

No	**Handshake Options**	**Memory Usage Bytes**	**Packet Size ClientHello**	**Packet Size ServerHello**	**Packet Size App Data**	**Packet Sent Size Bytes**	**Packet Recv Size Bytes**	**Time KeyGen Sec**	**Time Handshake Initiate Sec**	**Time Handshake Process Sec**
1	aesgcm-x2551-sha256	2858	162	146	152	82	77	0.008927	0.001	10.94496
2	aesgcm-x2551-sha512	2954	194	178	184	82	77	0	0	10.95506
3	aesgcm-x2551-blake2s	2859	162	146	152	82	77	0	0	10.95544
4	aesgcm-x2551-blake2b	2955	194	178	184	82	77	0	0	10.95625
5	chacha-x2551-sha256	3190	158	142	148	82	77	0.006152	0.001559	10.9473
6	chacha-x2551-sha512	3286	190	174	180	82	77	0	0	10.95591
7	chacha-x2551-blake2s	3191	158	142	148	82	77	0	0	10.95533
8	chacha-x2551-blake2b	3287	190	174	180	82	77	0	0	10.95591

Table 5.16: IN Pattern Handshake - Gateway (Server)

No	**Handshake Options**	**Memory Usage Bytes**	**Packet Size ClientHello**	**Packet Size ServerHello**	**Packet Size AppData**	**Packet Sent Size Bytes**	**Packet Recv Size Bytes**	**Time Handshake Sec**
1	aesgcm-x2551-sha256	5354	162	146	152	66	93	6.896478
2	aesgcm-x2551-sha512	5541	194	178	184	66	93	6.885847
3	aesgcm-x2551-blake2s	5407	162	146	152	66	93	6.9018
4	aesgcm-x2551-blake2b	5527	194	178	184	66	93	6.902655
5	chacha-x2551-sha256	6057	158	142	148	66	93	6.898648
6	chacha-x2551-sha512	6177	190	174	180	66	93	6.90165
7	chacha-x2551-blake2s	6043	158	142	148	66	93	6.901287
8	chacha-x2551-blake2b	6155	190	174	180	66	93	6.90019

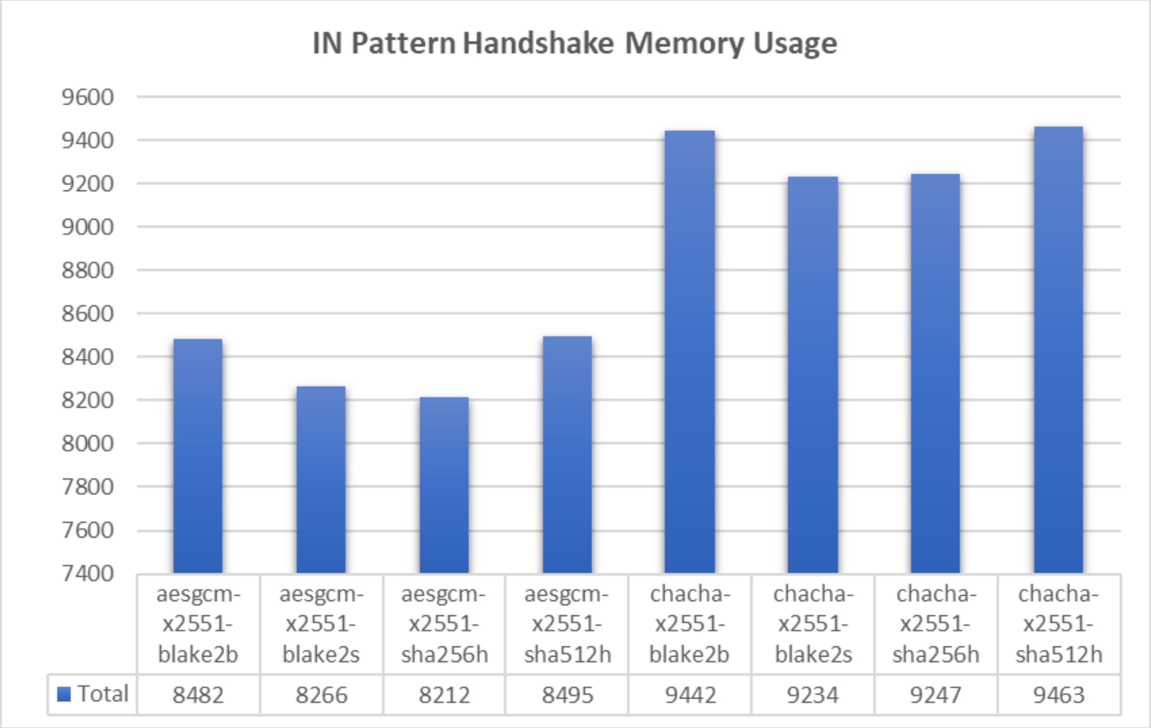


Figure 5.64: IN Handshake Memory Usage associated with different security options

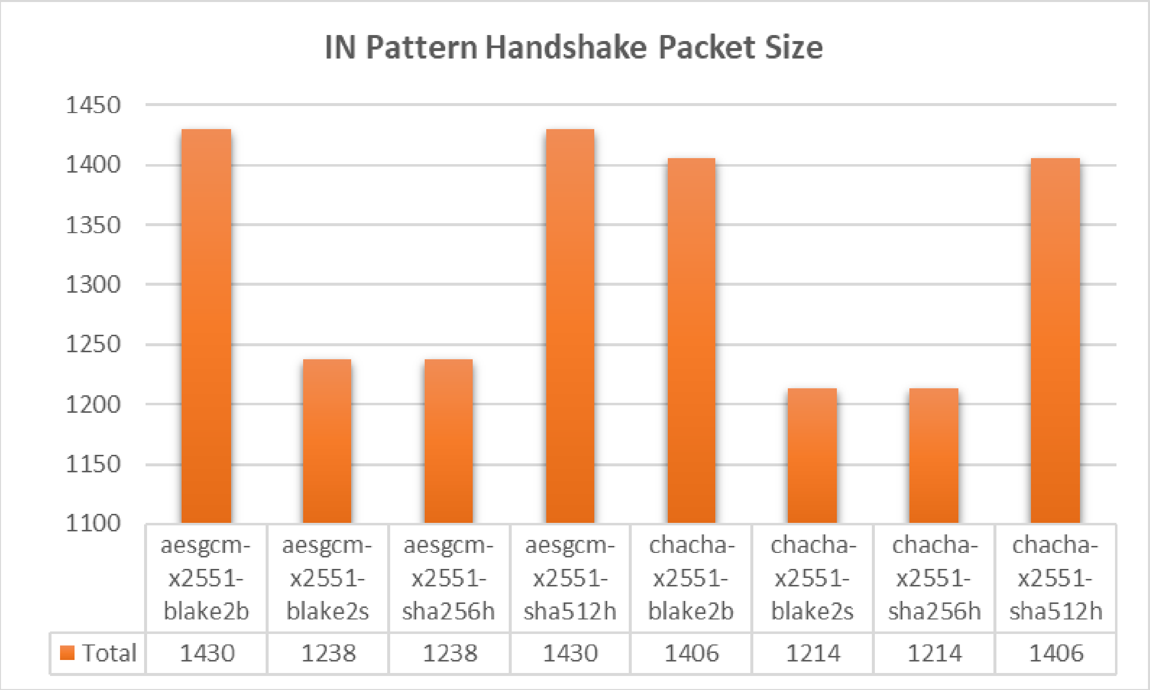


Figure 5.65: IN Handshake Packet Size associated with different security options

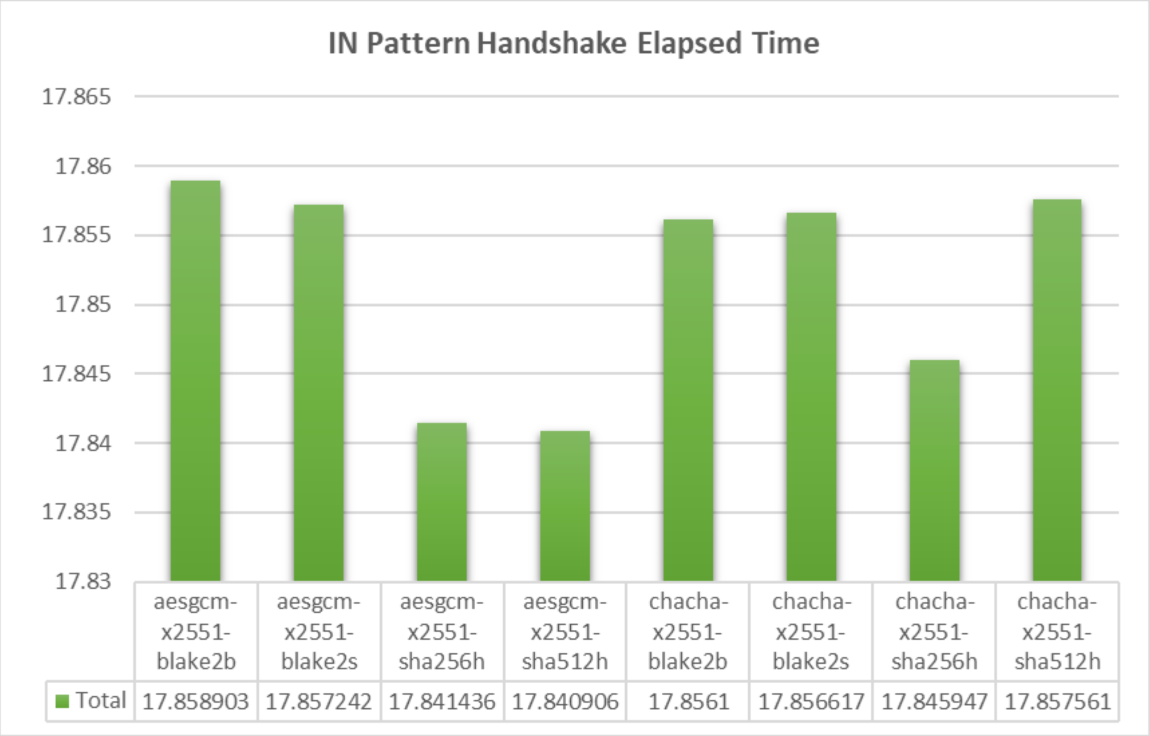


Figure 5.66: IN Handshake Elapsed Time associated with different security options

IX Pattern

This analysis was presented through two tables 5.17 and 5.18, detailing the process from the perspectives of both the meter and the gateway, and was followed by a chart that summarized the findings. This chart provided a clear overview of memory usage, packet size, and the time required to complete the handshake and transmit meter data.

1. **IX handshake memory usage** depicted in Figure 5.67 we can observe the following:
    - aesgcm-x2551-blake2b and aesgcm-x2551-blake2s are the most memory-efficient combinations, using the least memory.
    - chacha-x2551-blake2b, chacha-x2551-blake2s, and chacha-x2551-sha512 combinations show higher memory usage, with chacha-x2551-sha512 using the most memory overall.
2. **IX handshake packet size** depicted in Figure 5.68, we can observe the following:
    - The packet size varies significantly across combinations, with aesgcm-x2551-blake2b and aesgcm-x2551-sha512 having the largest packet sizes.
    - chacha-x2551-blake2s and chacha-x2551-sha256 have the smallest packet sizes, indicating potential efficiency in terms of data transmission.
3. **IX handshake elapsed time** depicted in Figure 5.69 we can observe the following:
    - The elapsed time for the handshake is the shortest for aesgcm-x2551-blake2b, suggesting a faster handshake process for this combination.
    - The chacha-x2551-sha512 combination shows a slightly longer elapsed time, which is consistent with its higher memory usage and packet size, possibly indicating a trade-off for using this combination.

In summary, the IX pattern graphs suggest that aesgcm-x2551-blake2b and aesgcm-x2551-blake2s may offer a balance between memory usage and handshake speed, while chacha-x2551-sha512 seems to be the most resource-intensive option. These observations can be crucial for optimizing handshake performance in cryptographic communications.

KK Pattern

This analysis was presented through two tables 5.19 and 5.20, detailing the process from the perspectives of both the meter and the gateway, and was followed by a chart that summarized the findings. This chart provided a clear overview of memory usage, packet size,

Table 5.17: IX Pattern Handshake - Meter (Client)

No	**Handshake Options**	**Memory Usage Bytes**	**Packet Size ClientHello**	**Packet Size ServerHello**	**Packet Size AppData**	**Packet Sent Size Bytes**	**Packet Recv Size Bytes**	**Time KeyGen Sec**	**Time Handshake Initiate Sec**	**Time Handshake Process Sec**
1	aesgcm-x2551-sha256	2858	162	194	152	82	125	0.008703	0.001	11.18598
2	aesgcm-x2551-sha512	2954	194	226	184	82	125	0	0	11.19514
3	aesgcm-x2551-blake2s	2859	162	194	152	82	125	0	0	11.19683
4	aesgcm-x2551-blake2b	2955	194	226	184	82	125	0	0	11.1791
5	chacha-x2551-sha256	3190	158	190	148	82	125	0.006392	0.001	11.18735
6	chacha-x2551-sha512	3286	190	222	180	82	125	0	0	11.18051
7	chacha-x2551-blake2s	3191	158	190	148	82	125	0	0	11.19498
8	chacha-x2551-blake2b	3287	190	222	180	82	125	0	0	11.19605

Table 5.18: IX Pattern Handshake - Gateway (Server)

Index	**Handshake Options**	**Memory Usage Bytes**	**Packet Size ClientHello**	**Packet Size ServerHello**	**Packet Size AppData**	**Packet Sent Size Bytes**	**Packet Recv Size Bytes**	**Time Handshake Initiate Sec**	**Time Handshake Process Sec**
1	aesgcm-x2551-sha256	5469	162	194	152	114	93	0	7.137603
2	aesgcm-x2551-sha512	5589	194	226	184	114	93	0	7.141325
3	aesgcm-x2551-blake2s	5455	162	194	152	114	93	0	7.127405
4	aesgcm-x2551-blake2b	5575	194	226	184	114	93	0	7.128213
5	chacha-x2551-sha256	6498	158	190	148	114	93	0	7.12363
6	chacha-x2551-sha512	6618	190	222	180	114	93	0	7.126549
7	chacha-x2551-blake2s	6484	158	190	148	114	93	0	7.125729
8	chacha-x2551-blake2b	6596	190	222	180	114	93	0	7.126559

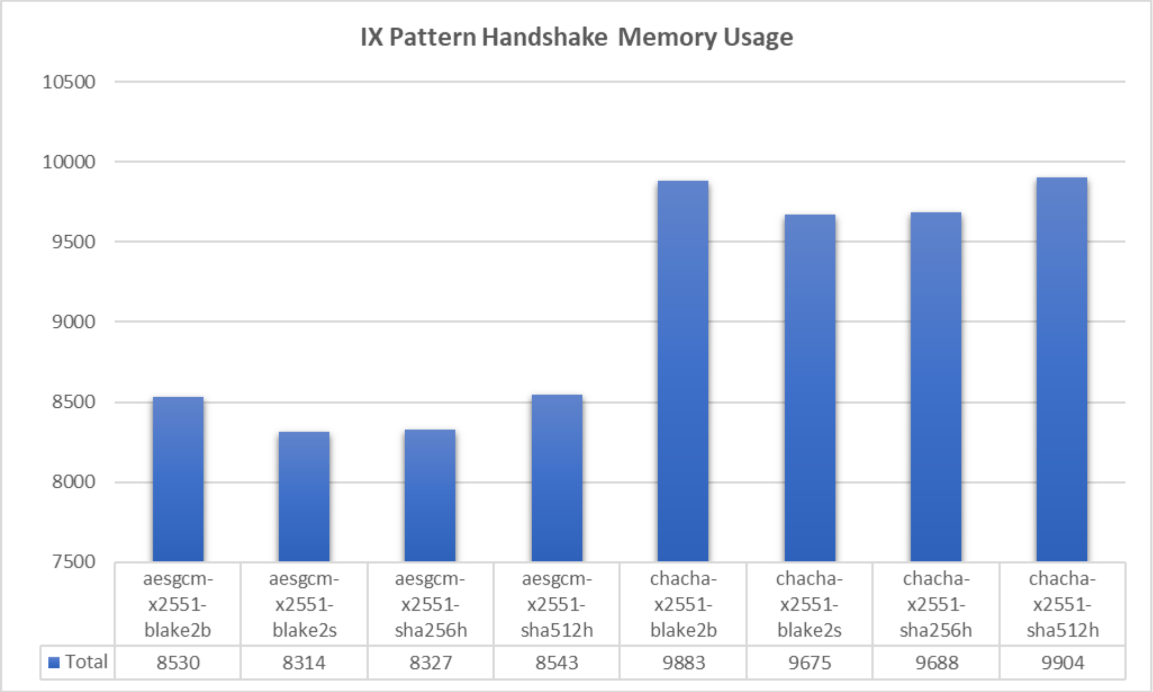


Figure 5.67: IX Handshake Memory Usage associated with different security options

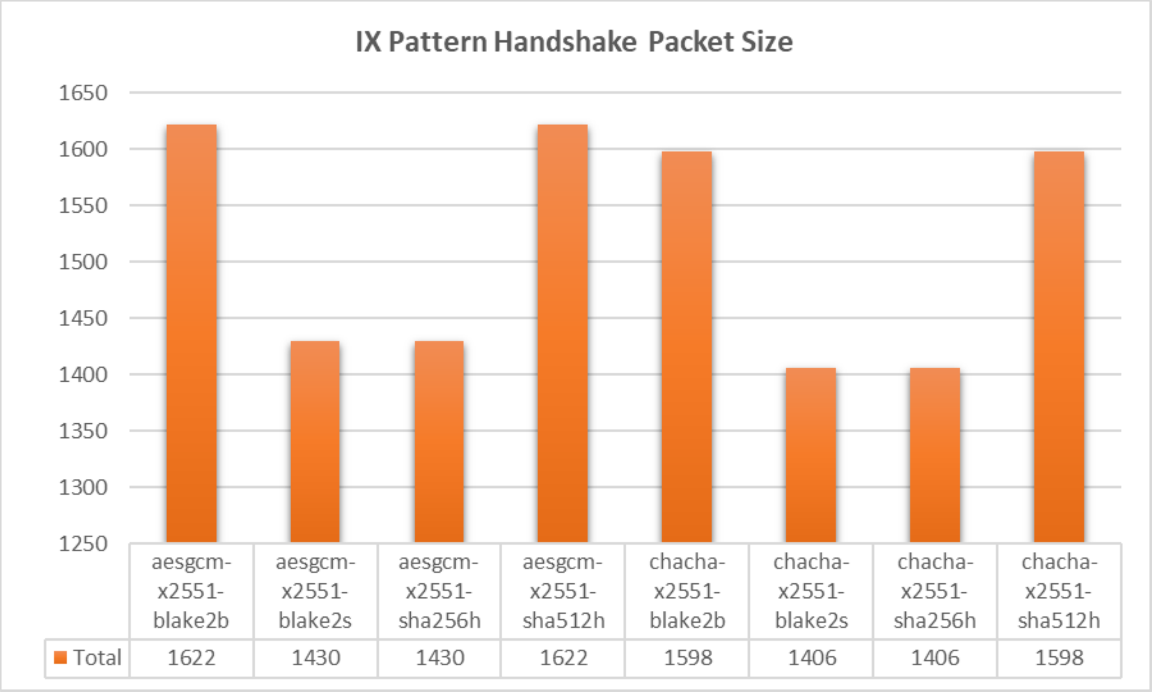


Figure 5.68: IX Handshake Packet Size associated with different security options

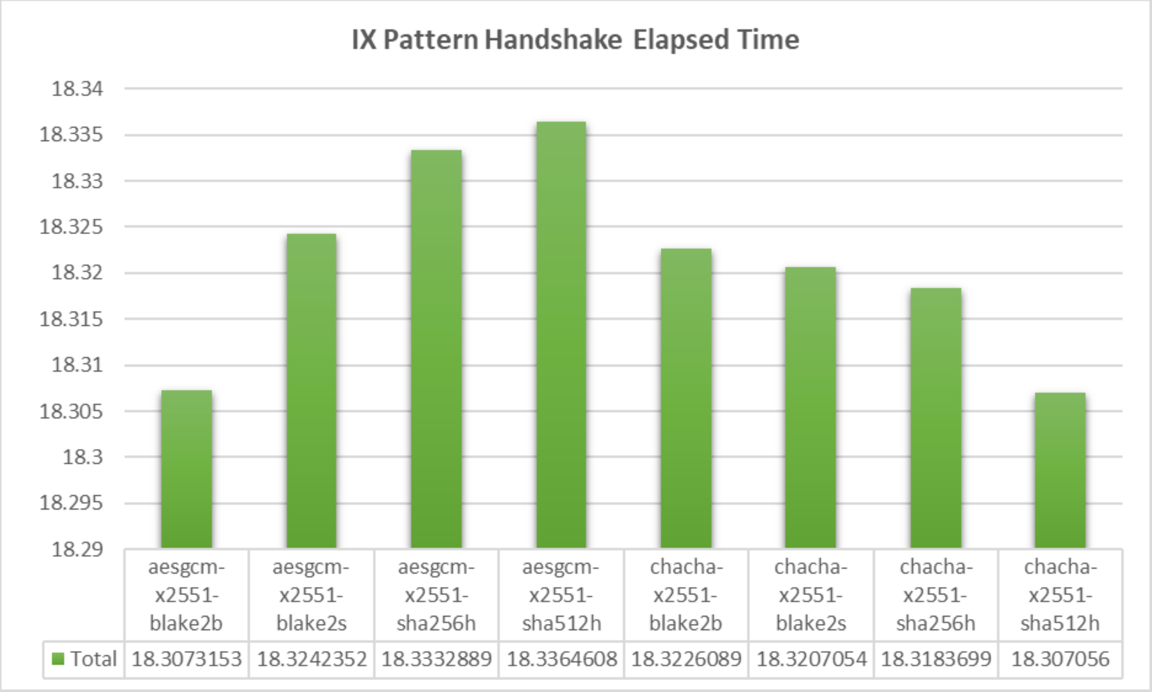


Figure 5.69: IX Handshake Elapsed Time associated with different security options

and the time required to complete the handshake and transmit meter data.

1. **KK handshake memory usage** depicted in Figure 5.70 we can observe the following:
    - The combination aesgcm-x2551-sha512 stands out with a significantly higher memory usage compared to the other combinations.
    - The chacha-x448-blake2b, chacha-x448-blake2s, chacha-x448-sha256, and chacha-x448-sha512 combinations show lower memory usage.
    - Memory usage for aesgcm with x2551 and Chacha with x2551 combinations are moderate and fairly consistent across the different hashing algorithms.
2. **KK** depicted in Figure 5.71, we can observe the following:
    - The packet sizes for aesgcm-x2551-blake2b, aesgcm-x448-sha512, and chacha-x448-sha512 are the largest, indicating a potential increase in bandwidth usage.
    - aesgcm-x2551-blake2s, aesgcm-x2551-sha256, chacha-x2551-blake2s, and chacha-x2551-sha256 have smaller packet sizes, which could be more bandwidth-efficient.
    - Overall, Chacha combinations with x448 tend to result in larger packet sizes than their x2551 counterparts.
3. **KK handshake elapsed time** depicted in Figure 5.72 we can observe the following:
    - There is a minimal variation in elapsed time across all combinations, with the majority falling between 17.7 to 18.2 seconds.
    - The aesgcm-x448-blake2b and chacha-x448-sha512 combinations have the longest elapsed times.
    - The aesgcm-x2551-blake2b combination has the shortest elapsed time, suggesting a slightly faster handshake process.

These observations suggest that the choice of security properties can significantly impact the memory and the speed of the handshake process. The aesgcm-x2551 combinations tend to be more efficient in terms of memory and packet size, while aesgcm-x448 and chacha-x448 combinations generally consume more resources. These metrics are crucial for assessing the performance of different cryptographic protocols within the KK pattern framework.

Table 5.19: KK Pattern Handshake - Meter (Client)

No	**Handshake Options**	**Memory Usage Bytes**	**Packet Size ClientHello**	**Packet Size ServerHello**	**Packet Size AppData**	**Packet Sent Size Bytes**	**Packet Recv Size Bytes**	**Time KeyGen Sec**	**Time Handshake Initiate Sec**	**Time Handshake Process Sec**
1	aesgcm-x2551-sha256	16470	146	146	152	66	77	0	0	10.89934
2	aesgcm-x2551-sha512	7633	178	178	184	66	77	0	0	10.87855
3	aesgcm-x2551-blake2s	6746	146	146	152	66	77	0	0	10.87801
4	aesgcm-x2551-blake2b	6162	178	178	184	66	77	0	0	10.87514
5	aesgcm-x448d-sha256	5770	194	194	176	90	101	0	0	11.11983
6	aesgcm-x448d-sha512	5562	226	226	208	90	101	0	0	11.11624
7	aesgcm-x448d-blake2s	5091	194	194	176	90	101	0	0	11.11647
8	aesgcm-x448d-blake2b	4859	226	226	208	90	101	0	0	11.11648
9	chacha-x2551-sha256	7057	142	142	148	66	77	0	0	10.87502
10	chacha-x2551-sha512	5240	174	174	180	66	77	0	0	10.87662
11	chacha-x2551-blake2s	4849	142	142	148	66	77	0	0	10.87505
12	chacha-x2551-blake2b	4777	174	174	180	66	77	0	0	10.87563
13	chacha-x448d-sha256	5190	190	190	172	90	101	0	0	11.11922
14	chacha-x448d-sha512	5022	222	222	204	90	101	0	0	11.11573
15	chacha-x448d-blake2s	4639	190	190	172	90	101	0.00177	0.002143	11.10254
16	chacha-x448d-blake2b	4527	222	222	204	90	101	0	0	11.11604

Table 5.20: KK Pattern Handshake - Gateway (Server)

No	**Handshake Options**	**Memory Usage Bytes**	**Packet Size ClientHello**	**Packet Size ServerHello**	**Packet Size AppData**	**Packet Calculated Size Bytes**	**Packet Sent Size Bytes**	**Packet Recv Size Bytes**	**Time Handshake Initiate Sec**	**Time Handshake Process Sec**
1	aesgcm-x2551-sha256	25509	146	146	152	146	66	77	0	6.909058
2	aesgcm-x2551-sha512	9850	178	178	184	178	66	77	0	6.89177
3	aesgcm-x2551-blake2s	9204	146	146	152	146	66	77	0	6.893079
4	aesgcm-x2551-blake2b	8812	178	178	184	178	66	77	0	6.891184
5	aesgcm-x448d-sha256	7705	194	194	176	194	90	101	0.005238	7.001273
6	aesgcm-x448d-sha512	7585	226	226	208	226	90	101	0.004997	7.006792
7	aesgcm-x448d-blake2s	7275	194	194	176	194	90	101	0	7.02062
8	aesgcm-x448d-blake2b	7163	226	226	208	226	90	101	0.015625	7.017461
9	chacha-x2551-sha256	10034	142	142	148	142	66	77	0	6.893302
10	chacha-x2551-sha512	8313	174	174	180	174	66	77	0	6.891253
11	chacha-x2551-blake2s	7987	142	142	148	142	66	77	0.015625	6.892152
12	chacha-x2551-blake2b	7915	174	174	180	174	66	77	0	6.89267
13	chacha-x448d-sha256	7925	190	190	172	190	90	101	0.004997	7.001492
14	chacha-x448d-sha512	7845	222	222	204	222	90	101	0.003998	6.990957
15	chacha-x448d-blake2s	7535	190	190	172	190	90	101	0	7.016986
16	chacha-x448d-blake2b	10353	222	222	204	222	90	101	0.015625	7.002285

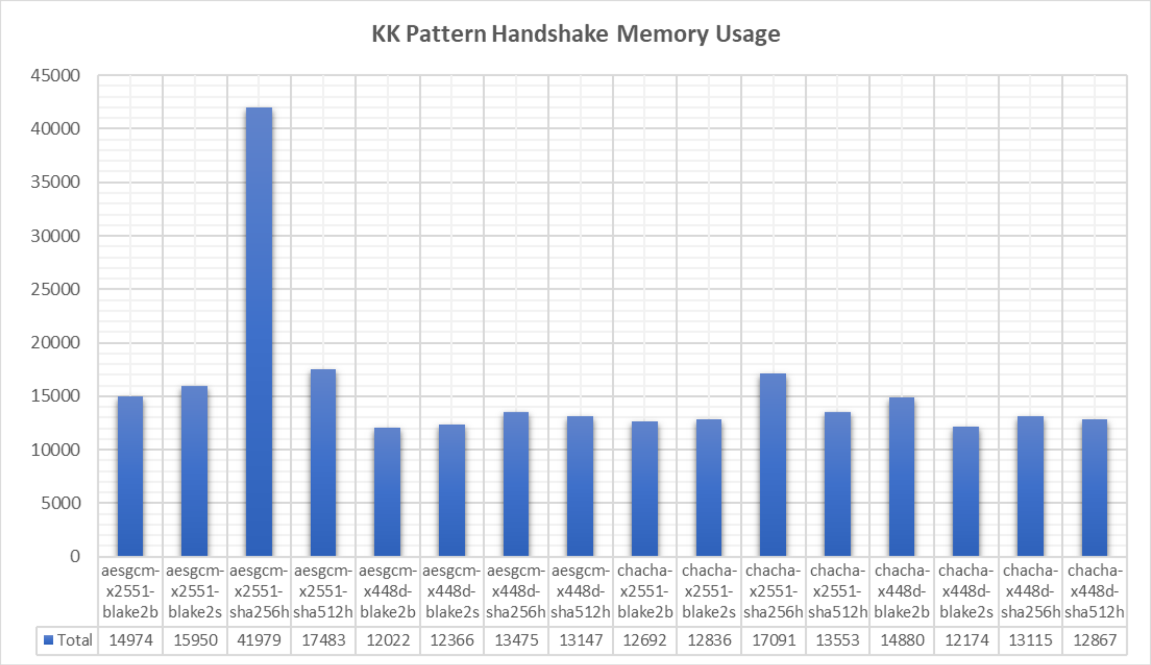


Figure 5.70: KK Handshake Memory Usage associated with different security options

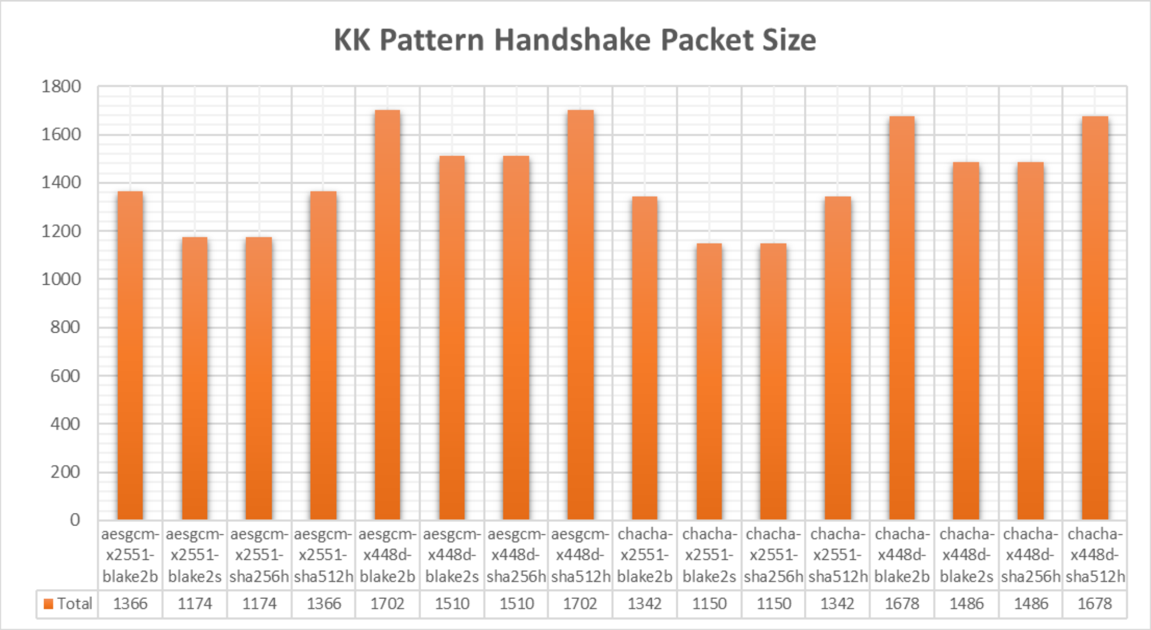


Figure 5.71: KK Handshake Packet Size associated with different security options

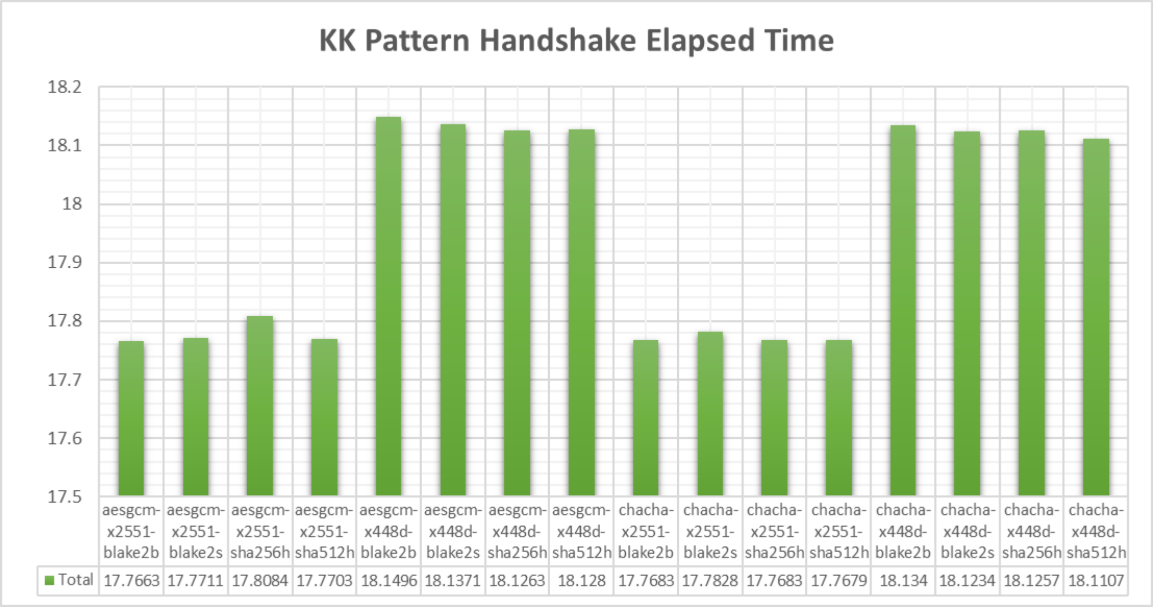


Figure 5.72: KK Handshake Memory Usage associated with different security options

KN Pattern

This analysis was presented through two tables 5.21 and 5.22, detailing the process from the perspectives of both the meter and the gateway, and was followed by a chart that summarized the findings. This chart provided a clear overview of memory usage, packet size, and the time required to complete the handshake and transmit meter data.

1. **KN handshake memory usage** depicted in Figure 5.73 we can observe the following:
    - Memory usage is relatively consistent across the different combinations, with most falling between 8800 and 9600 units (likely kilobytes).
    - The aesgcm-x2551-sha512 and chacha-x448-sha512 combinations have the highest memory usage, while aesgcm-x2551-blake2s uses the least.
2. **KN handshake packet size** depicted in Figure 5.74 we can observe the following:
    - The packet sizes show more variance, with aesgcm-x448-sha512 and chacha-x448-sha512 having the largest packet sizes, which might affect bandwidth usage.
    - aesgcm-x2551-blake2s and chacha-x2551-blake2s display the smallest packet sizes, potentially offering better bandwidth efficiency.

Table 5.21: KN Pattern Handshake - Meter (Client)

No	**Handshake Options**	**Memory Usage Bytes**	**Packet Size ClientHello**	**Packet Size ServerHello**	**Packet Size AppData**	**Packet Sent Size Bytes**	**Packet Recv Size Bytes**	**Time KeyGen Sec**	**Time Handshake Initiate Sec**	**Time Handshake Process Sec**
1	aesgcm-x2551-sha256	3178	130	146	152	50	77	0.010633	0.000998	10.78349
2	aesgcm-x2551-sha512	3226	162	178	184	50	77	0	0	10.79485
3	aesgcm-x2551-blake2s	3099	130	146	152	50	77	0	0	10.79615
4	aesgcm-x2551-blake2b	3163	162	178	184	50	77	0	0	10.79527
5	aesgcm-x448d-sha256	3208	178	194	176	74	101	0	0	11.03827
6	aesgcm-x448d-sha512	3264	210	226	208	74	101	0	0	11.02241
7	aesgcm-x448d-blake2s	3137	178	194	176	74	101	0	0	11.01952
8	aesgcm-x448d-blake2b	3169	210	226	208	74	101	0	0	11.0353
9	chacha-x2551-sha256	3214	126	142	148	50	77	0	0	10.79261
10	chacha-x2551-sha512	3278	158	174	180	50	77	0	0	10.79593
11	chacha-x2551-blake2s	3167	126	142	148	50	77	0	0	10.79654
12	chacha-x2551-blake2b	3255	158	174	180	50	77	0.000999	0.000999	10.78997
13	chacha-x448d-sha256	3324	174	190	172	74	101	0.001	0.000999	11.03068
14	chacha-x448d-sha512	3420	206	222	204	74	101	0	0	11.03662
15	chacha-x448d-blake2s	3325	174	190	172	74	101	0	0	11.03523
16	chacha-x448d-blake2b	3421	206	222	204	74	101	0	0	11.03656

3. **KN handshake elapsed time** depicted in Figure 5.75 we can observe the following:
    - The elapsed time for the handshake process is relatively close across the board, with a range between roughly 17.7 to 18.1 seconds.
    - The aesgcm-x448-blake2b combination exhibits the longest elapsed time, while aesgcm-x2551-blake2b has the shortest, indicating a marginal speed advantage.

In conclusion, the KN Pattern shows a relatively even distribution of memory usage across most algorithm combinations, with specific sha512 variants using more memory. Packet sizes and handshake times also vary but remain within a narrow range, suggesting that the differences in performance between these combinations are modest. These factors are important when considering the balance between security and performance in cryptographic protocols.

KX Pattern

This analysis was presented through two tables 5.23 and 5.24, detailing the process from the perspectives of both the meter and the gateway, and was followed by a chart that summarized the findings. This chart provided a clear overview of memory usage, packet size, and the time required to complete the handshake and transmit meter data.

1. **KX handshake memory usage** depicted in Figure 5.76 we can observe the following:
    - The memory usage varies modestly among the different combinations, with the lowest usage seen in aesgcm-x2551-blake2b and the highest in chacha-x2551-

Table 5.22: KN Pattern Handshake - Gateway (Server)

No	Handshake Options	Memory Usage Bytes	Packet Size ClientHello	Packet Size ServerHello	Packet size AppData	Packet Sent Size Bytes	Packet Recv Size Bytes	Time Handshake Initiate Sec	Time Handshake Process Sec
1	aesgcm-x2551-sha256	5988	130	146	152	66	61	0	6.893144
2	aesgcm-x2551-sha512	6012	162	178	184	66	61	0	6.898363
3	aesgcm-x2551-blake2s	5782	130	146	152	66	61	0	6.891647
4	aesgcm-x2551-blake2b	5798	162	178	184	66	61	0	6.891198
5	aesgcm-x448d-sha256	5880	178	194	176	90	85	0	7.017592
6	aesgcm-x448d-sha512	5912	210	226	208	90	85	0.001	7.003622
7	aesgcm-x448d-blake2s	5682	178	194	176	90	85	0	7.002288
8	aesgcm-x448d-blake2b	5674	210	226	208	90	85	0	7.017092
9	chacha-x2551-sha256	6008	126	142	148	66	61	0	6.892022
10	chacha-x2551-sha512	6056	158	174	180	66	61	0	6.892097
11	chacha-x2551-blake2s	5898	126	142	148	66	61	0	6.895557
12	chacha-x2551-blake2b	6002	158	174	180	66	61	0.000998	6.891687
13	chacha-x448d-sha256	6044	174	190	172	90	85	0	7.019348
14	chacha-x448d-sha512	6140	206	222	204	90	85	0	7.024184
15	chacha-x448d-blake2s	5998	174	190	172	90	85	0	7.021504
16	chacha-x448d-blake2b	6102	206	222	204	90	85	0	7.032141

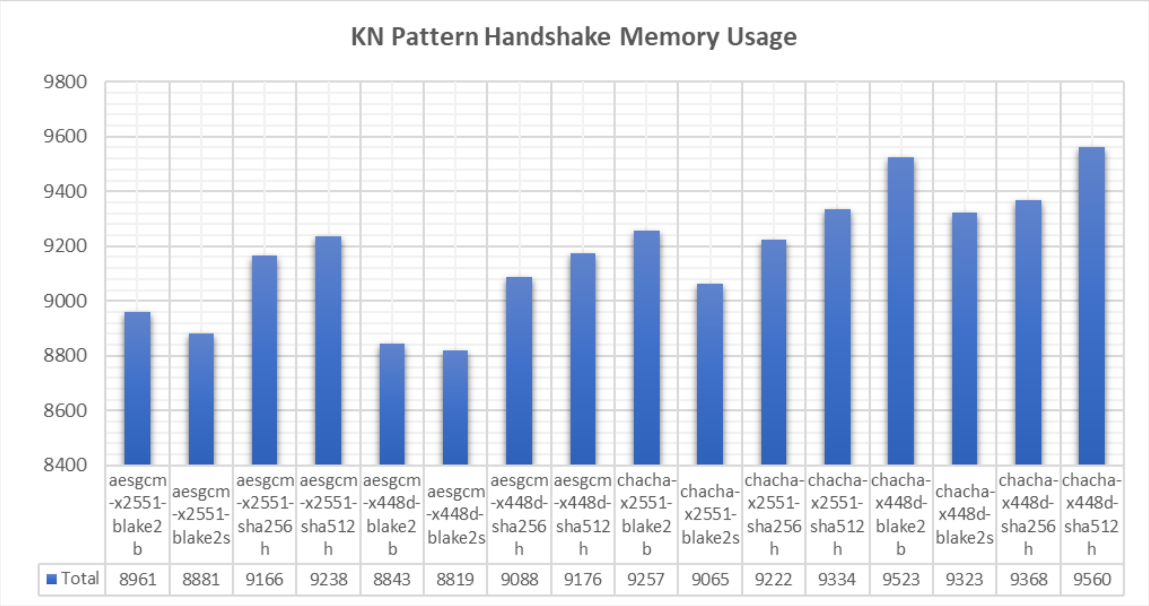


Figure 5.73: KN Handshake Memory Usage associated with different security options

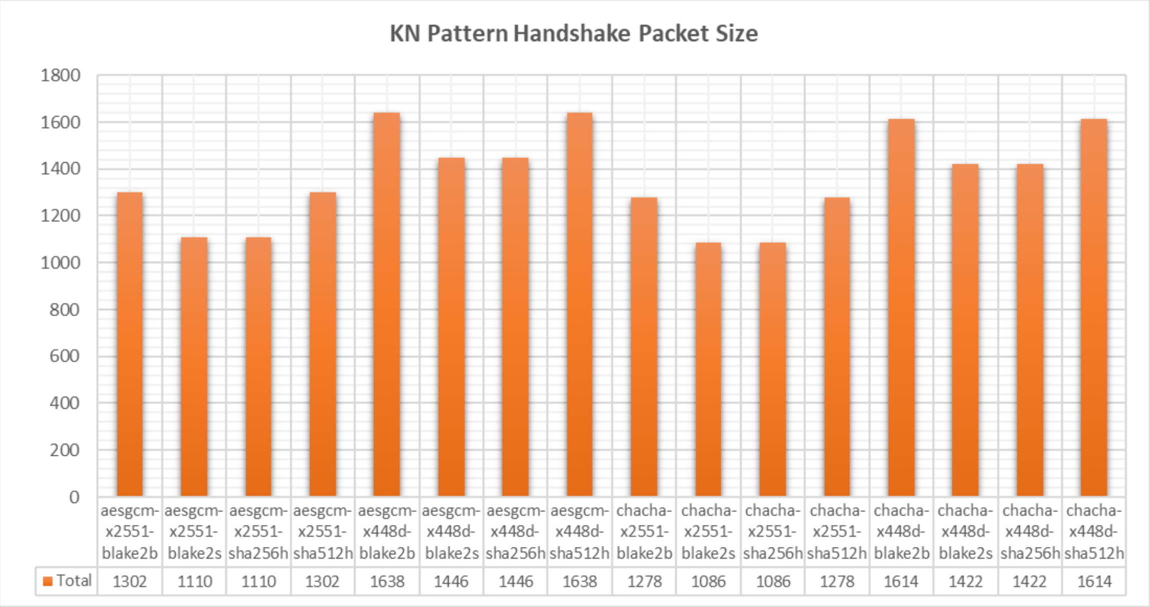


Figure 5.74: KN Handshake Packet Size associated with different security options

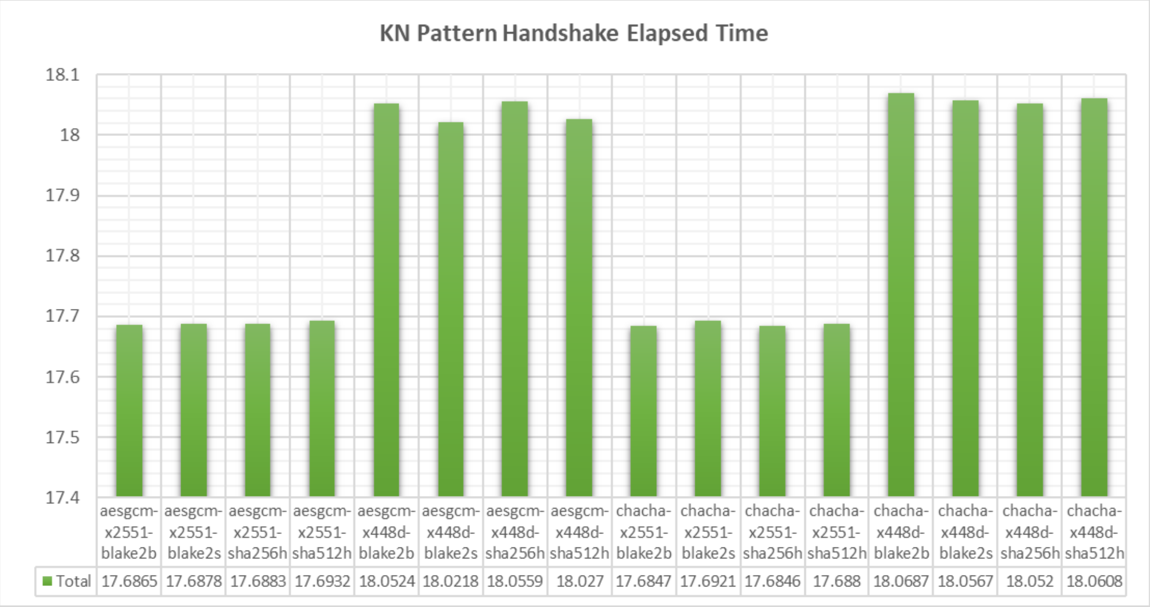


Figure 5.75: Enter Caption

Table 5.23: KX Pattern Handshake - Meter (Client)

No	**Handshake Options**	**Memory Usage Bytes**	**Packet Size ClientHello**	**Packet Size ServerHello**	**Packet size AppData**	**Packet Sent Size Bytes**	**Packet Recv Size Bytes**	**Time Key Gen Sec**	**Time Handshake Initiate Sec**	**Time Handshake Process Sec**
1	aesgcm-x2551-sha256	2826	130	194	152	50	125	0	0.008672	11.02698
2	aesgcm-x2551-sha512	2922	162	226	184	50	125	0	0	11.03556
3	aesgcm-x2551-blake2s	2827	130	194	152	50	125	0	0	11.03634
4	aesgcm-x2551-blake2b	2923	162	226	184	50	125	0	0	11.03149
5	chacha-x2551-sha256	3158	126	190	148	50	125	0.006357	0.001001	11.02817
6	chacha-x2551-sha512	3254	158	222	180	50	125	0	0.000999	11.03088
7	chacha-x2551-blake2s	3159	126	190	148	50	125	0	0	11.03478
8	chacha-x2551-blake2b	3255	158	222	180	50	125	0	0.006003	11.01746

sha512.

- The chacha-x2551 combinations generally use more memory than aesgcm-x2551 combinations, with sha512 hash showing the highest memory usage in both aesgcm and Chacha.

2. **KX handshake packet size** depicted in Figure 5.77, we can observe the following:
   - The packet size for aesgcm-x2551-sha512 is notably larger than the other combinations, suggesting a higher bandwidth requirement.
   - aesgcm-x2551-blake2s, chacha-x2551-blake2s, and chacha-x2551-sha256 have smaller packet sizes, which could be advantageous for bandwidth conservation.
   - Similar to the memory usage, chacha-x2551 combinations tend to result in larger packet sizes than aesgcm-x2551 combinations.
3. **KX handshake elapsed time** depicted in Figure 5.78 we can observe the following:
   - The elapsed times are very close, with all falling within the 18.16 to 18.17 seconds range, indicating minimal variance in handshake duration across the algorithm combinations.
   - aesgcm-x2551-sha512 shows the longest elapsed time, while aesgcm-x2551-blake2b and aesgcm-x2551-blake2s are among the shortest, although the differences are slight.

In summary, the KX Pattern demonstrates slight differences in resource usage among the cryptographic combinations. The chacha-x2551-sha512 combination tends to use the most resources in terms of memory and packet size, while aesgcm-x2551-blake2b and aesgcm-x2551-blake2s are more resource-efficient.

Table 5.24: KX Pattern Handshake - Gateway (Server)

No	**Handshake Options**	**Memory Usage Bytes**	**Packet Size ClientHello**	**Packet Size ServerHello**	**Packet Size AppData**	**Packet Sent Size Bytes**	**Packet Recv Size Bytes**	**Time Handshake Initiate Sec**	**Time Handshake Process Sec**
1	aesgcm-x2551-sha256h	5340	130	194	152	114	61	0	7.127288
2	aesgcm-x2551-sha512h	5452	162	226	184	114	61	0	7.137314
3	aesgcm-x2551-blake2s	5310	130	194	152	114	61	0	7.126678
4	aesgcm-x2551-blake2b	5355	162	226	184	114	61	0	7.13091
5	chacha-x2551-sha256h	6337	126	190	148	114	61	0.001999	7.130781
6	chacha-x2551-sha512h	6441	158	222	180	114	61	0	7.136995
7	chacha-x2551-blake2s	6299	126	190	148	114	61	0	7.131626
8	chacha-x2551-blake2b	6403	158	222	180	114	61	0	7.138586

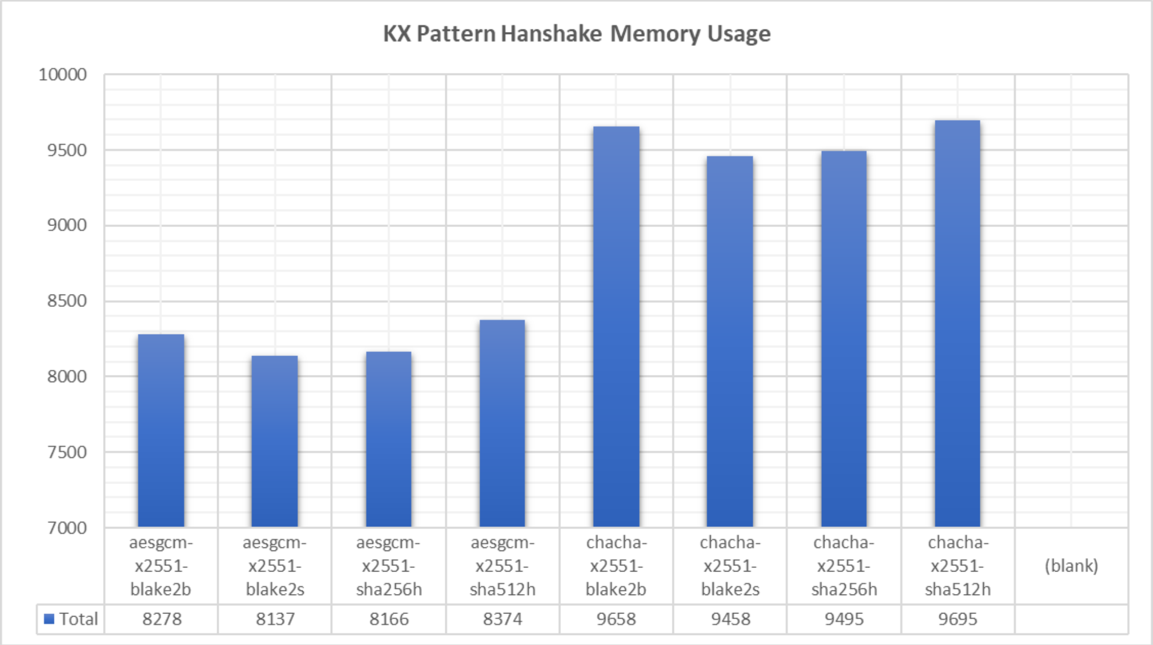


Figure 5.76: KX Handshake Memory Usage associated with different security options

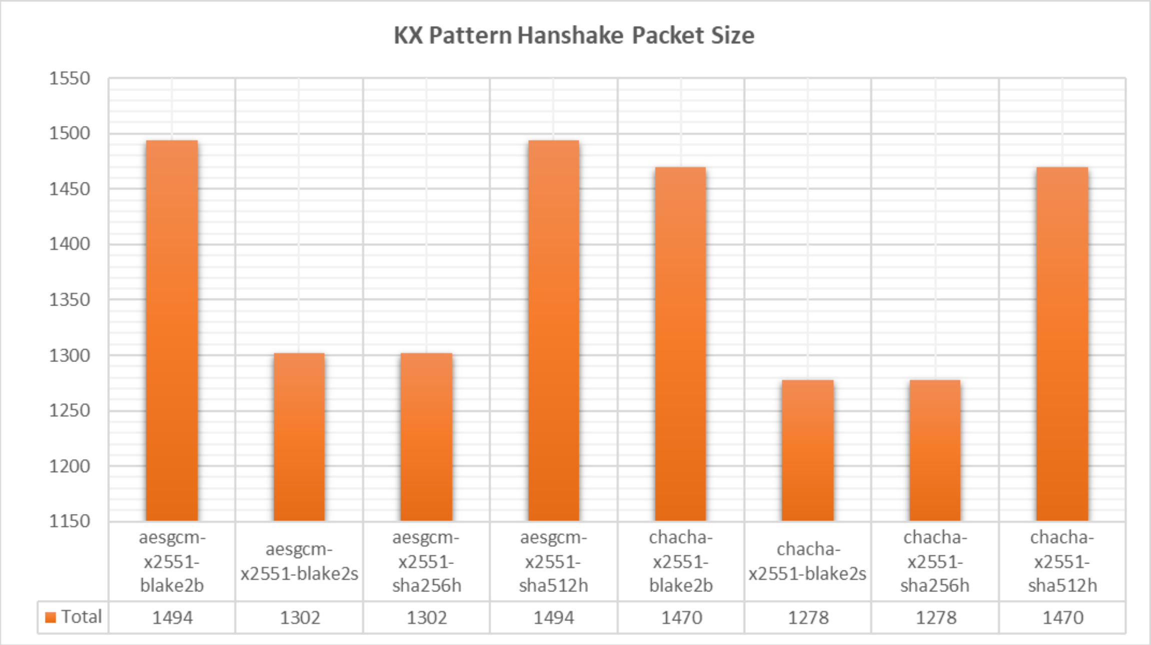


Figure 5.77: KX Handshake Packet Size associated with different security options

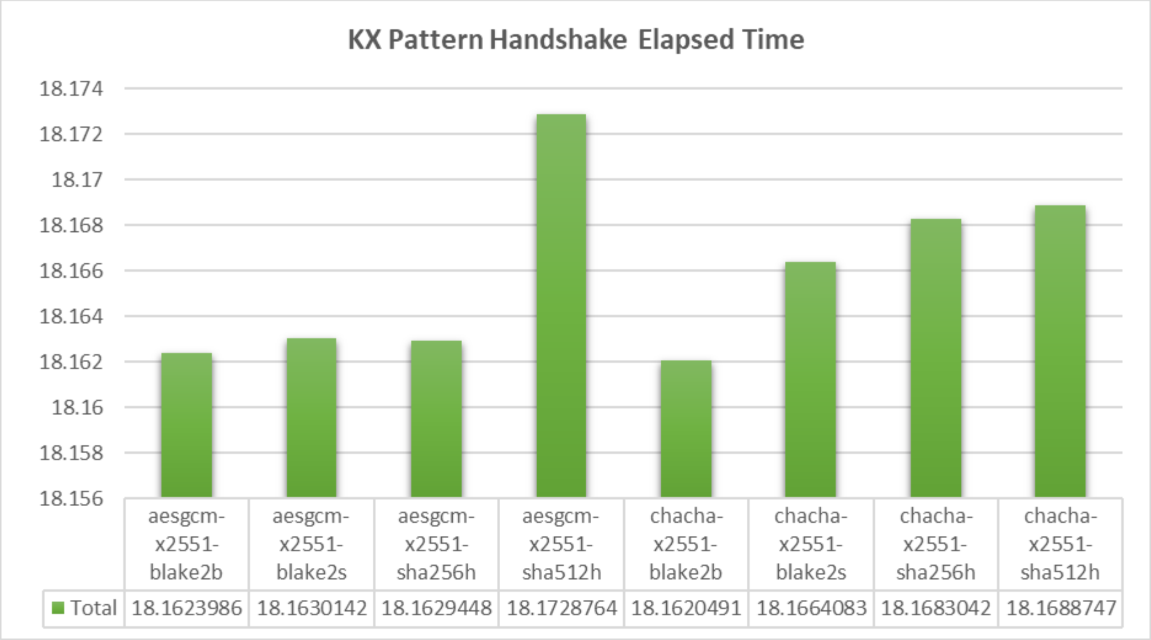


Figure 5.78: KX Handshake Elapsed Time associated with different security options

NK Pattern

This analysis was presented through two tables 5.25 and 5.26, detailing the process from the perspectives of both the meter and the gateway, and was followed by a chart that summarized the findings. This chart provided a clear overview of memory usage, packet size, and the time required to complete the handshake and transmit meter data.

1. **NK handshake memory usage** depicted in Figure 5.79 we can observe the following:
    - The memory usage is uniformly distributed among the different combinations, ranging from the lowest at approximately 8200 to the highest at just over 10000 units (likely kilobytes).
    - chacha-x448 combinations consistently have higher memory usage compared to aesgcm-x2551 and aesgcm-x448 combinations.
2. **NK handshake packet size** depicted in Figure 5.80, we can observe the following:
    - The packet size varies with the highest recorded for aesgcm-x448-sha512 and chacha-x448-sha512, indicating potentially higher bandwidth consumption for these combinations.
    - aesgcm-x2551-blake2s, chacha-x2551-blake2s, and chacha-x2551-sha256 show smaller packet sizes, which could suggest better bandwidth efficiency.
3. **NK handshake elapsed time** depicted in Figure 5.81 we can observe the following:
    - The elapsed time for the handshake process shows minimal variance, with most combinations completing the handshake in around 17.7 to 18.1 seconds.
    - The aesgcm-x448-sha512 and chacha-x448-sha512 combinations have the longest elapsed times, which correlates with their higher memory and packet size usage.
    - The aesgcm-x2551-blake2b and aesgcm-x2551-blake2s combinations have some of the shortest elapsed times, indicating a slight efficiency in processing speed.

Overall, the NK Pattern suggests that the chacha-x448 combinations tend to consume more resources in terms of memory usage and potentially bandwidth, while aesgcm-x2551 combinations are generally more resource-efficient. These insights can be important for selecting cryptographic protocols in systems where resource constraints are a critical factor.

Table 5.25: NK Pattern Handshake - Meter (Client)

No	Handshake Options	Memory Usage Bytes	Packet Size ClientHello	Packet Size ServerHello	Packet Size AppData	Packet Sent Size Bytes	Packet Recv Size Bytes	Time KeyGen Sec	Time Handshake Initiate Sec	Time Handshake Process Sec
1	aesgcm-x2551-sha256	2972	146	146	152	66	77	0.008898	0.000998	10.87853
2	aesgcm-x2551-sha512	3100	178	178	184	66	77	0	0	10.8606
3	aesgcm-x2551-blake2s	2973	146	146	152	66	77	0	0	10.87473
4	aesgcm-x2551-blake2b	3101	178	178	184	66	77	0	0	10.8763
5	aesgcm-x448d-sha256	3138	194	194	176	90	101	0	0	11.11474
6	aesgcm-x448d-sha512	3266	226	226	208	90	101	0	0	11.10315
7	aesgcm-x448d-blake2s	3139	194	194	176	90	101	0	0	11.09977
8	aesgcm-x448d-blake2b	3267	226	226	208	90	101	0	0	11.10008
9	chacha-x2551-sha256	3632	142	142	148	66	77	0	0	10.87336
10	chacha-x2551-sha512	3760	174	174	180	66	77	0	0	10.87593
11	chacha-x2551-blake2s	3633	142	142	148	66	77	0	0	10.87641
12	chacha-x2551-blake2b	3761	174	174	180	66	77	0	0	10.87592
13	chacha-x448d-sha256	3798	190	190	172	90	101	0	0	11.11759
14	chacha-x448d-sha512	3926	222	222	204	90	101	0	0	11.11633
15	chacha-x448d-blake2s	3799	190	190	172	90	101	0	0	11.11558
16	chacha-x448d-blake2b	3927	222	222	204	90	101	0	0	11.11632

Table 5.26: NK Pattern Handshake - Gateway (Server)

Index	Handshake Options	Memory Usage Bytes	Packet Size ClientHello	Packet Size ServerHello	Packet Size AppData	Packet Sent Size Bytes	Packet Recv Size Bytes	Time Handshake Initiate Sec	Time Handshake Process Sec
1	aesgcm-x2551-sha256	5317	146	146	152	66	77	0.004	6.891664
2	aesgcm-x2551-sha512	5429	178	178	184	66	77	0	6.885021
3	aesgcm-x2551-blake2s	5228	146	146	152	66	77	0	6.898977
4	aesgcm-x2551-blake2b	5415	178	178	184	66	77	0	6.899714
5	aesgcm-x448d-sha256	5497	194	194	176	90	101	0	7.011044
6	aesgcm-x448d-sha512	5609	226	226	208	90	101	0	6.999236
7	aesgcm-x448d-blake2s	5467	194	194	176	90	101	0	7.011499
8	aesgcm-x448d-blake2b	5571	226	226	208	90	101	0	7.011703
9	chacha-x2551-sha256	6233	142	142	148	66	77	0	6.897406
10	chacha-x2551-sha512	6353	174	174	180	66	77	0	6.895652
11	chacha-x2551-blake2s	6219	142	142	148	66	77	0	6.899644
12	chacha-x2551-blake2b	6339	174	174	180	66	77	0	6.883427
13	chacha-x448d-sha256	6389	190	190	172	90	101	0	7.013474
14	chacha-x448d-sha512	6493	222	222	204	90	101	0.015624	7.012379
15	chacha-x448d-blake2s	6351	190	190	172	90	101	0.015623	7.01158
16	chacha-x448d-blake2b	6455	222	222	204	90	101	0.015626	7.012294

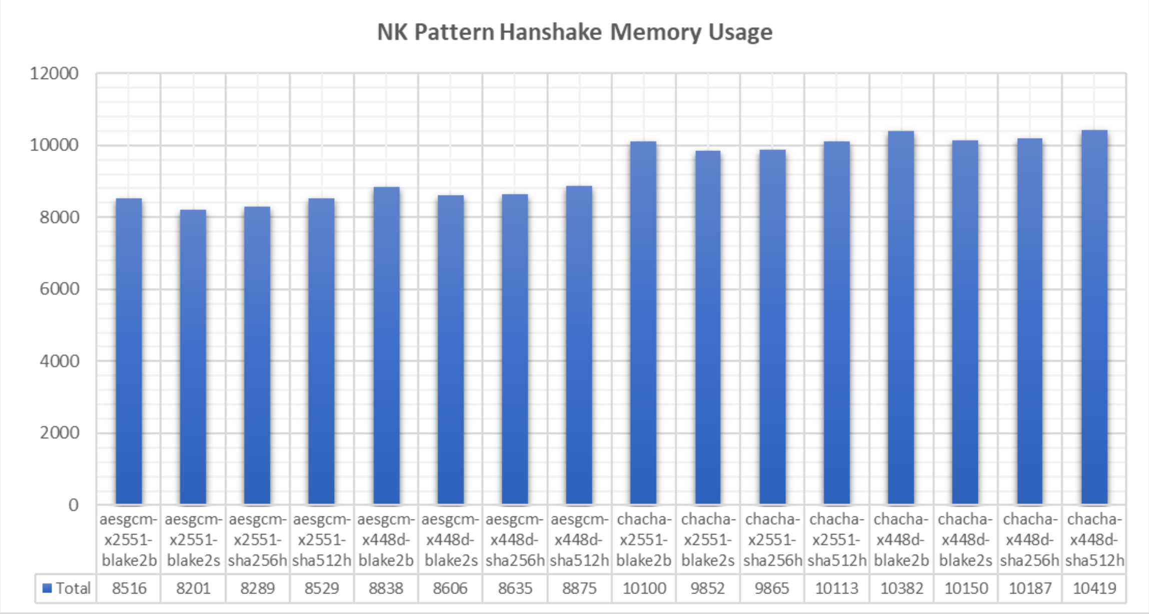


Figure 5.79: NK Handshake Memory Usage associated with different security options

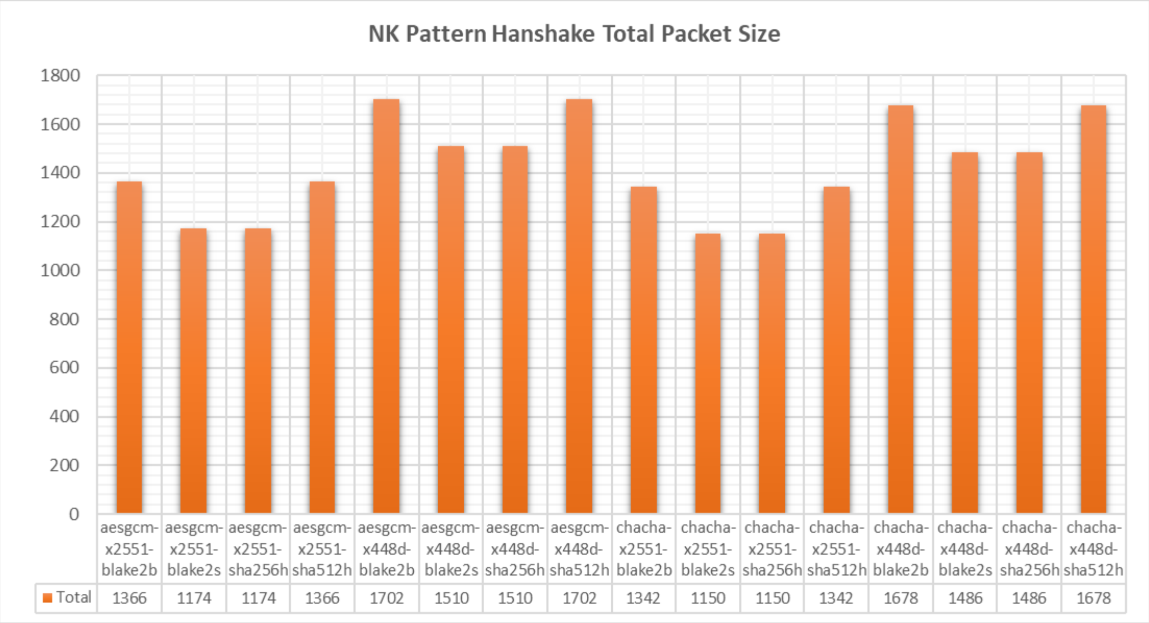


Figure 5.80: NK Handshake Packet Size associated with different security options

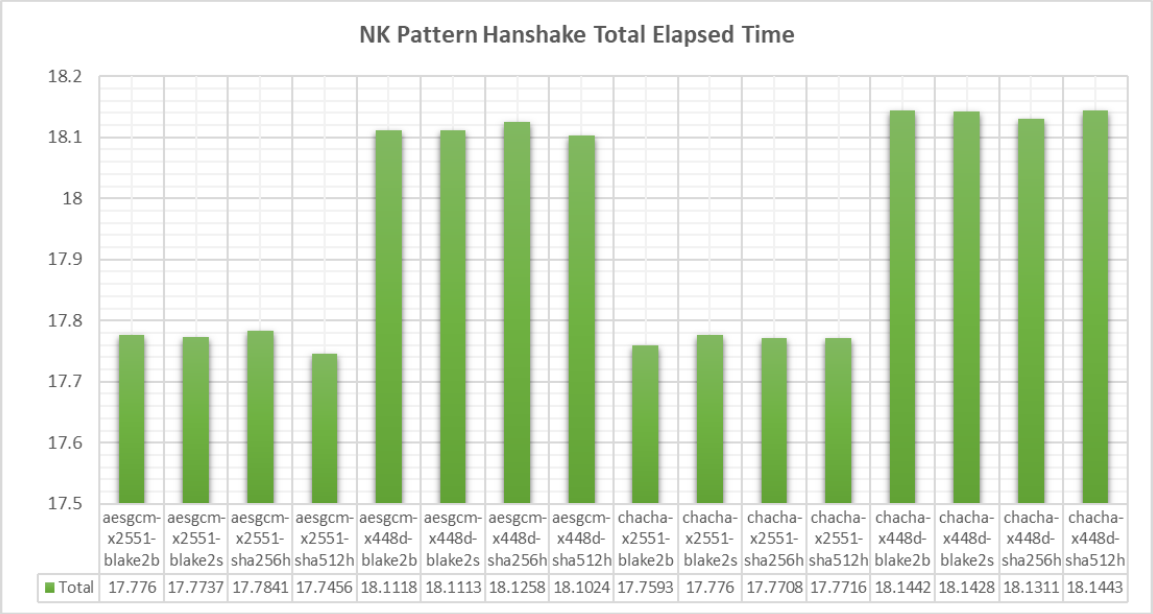


Figure 5.81: NK Handshake Elapsed Time associated with different security options

NN Pattern

This analysis was presented through two tables 5.27 and 5.28, detailing the process from the perspectives of both the meter and the gateway, and was followed by a chart that summarized the findings. This chart provided a clear overview of memory usage, packet size, and the time required to complete the handshake and transmit meter data.

1. **NN handshake memory usage** depicted in Figure 5.82 we can observe the following:
    - Memory usage during the handshake process shows significant differences between the algorithm combinations.
    - Combinations using chacha20 tend to have higher memory usage across both x25519 and x448, possibly indicating that chacha20 requires more memory resources.
    - The sha512 hashing function seems to consistently result in higher memory usage compared to sha256, across different encryption methods, which aligns with the larger output size of sha512.
2. **NN handshake packet size** depicted in Figure 5.83, we can observe the following:
    - The total packet size graph shows a more pronounced variation between differ-

Table 5.27: NN Pattern Handshake - Meter (Client)

No	**Handshake Options**	**Memory Usage Bytes**	**Packet Size ClientHello**	**Packet Size ServerHello**	**Packet Size AppData**	**Packet Sent Size Bytes**	**Packet Recv Size Bytes**	**Time KeyGen Sec**	**Time Handshake Initiate Sec**	**Time Handshake Process Sec**
1	aesgcm-x2551-sha256	2826	130	146	152	50	77	0.007382	0.000999	10.78874
2	aesgcm-x2551-sha512	2922	162	178	184	50	77	0	0	10.79467
3	aesgcm-x2551-blake2s	2827	130	146	152	50	77	0	0	10.79706
4	aesgcm-x2551-blake2b	2923	162	178	184	50	77	0	0	10.79538
5	aesgcm-x448d-sha256	2992	178	194	176	74	101	0	0	11.02224
6	aesgcm-x448d-sha512	3088	210	226	208	74	101	0	0	11.03467
7	aesgcm-x448d-blake2s	2993	178	194	176	74	101	0	0	11.03817
8	aesgcm-x448d-blake2b	3089	210	226	208	74	101	0	0	11.03388
9	chacha-x2551-sha256	3158	126	142	148	50	77	0	0	10.7961
10	chacha-x2551-sha512	3254	158	174	180	50	77	0	0	10.79566
11	chacha-x2551-blake2s	3159	126	142	148	50	77	0	0	10.79565
12	chacha-x2551-blake2b	3255	158	174	180	50	77	0	0	10.79666
13	chacha-x448d-sha256	3324	174	190	172	74	101	0	0	11.02215
14	chacha-x448d-sha512	3420	206	222	204	74	101	0	0	11.02004
15	chacha-x448d-blake2s	3325	174	190	172	74	101	0	0	11.01983
16	chacha-x448d-blake2b	3421	206	222	204	74	101	0	0	11.03459

ent algorithm combinations than the elapsed time graph.

- The x448 combinations generally result in larger packet sizes compared to x25519, which could be due to the larger key sizes associated with x448.
- There is a noticeable increase in packet size when sha512 is used as the hashing function compared to sha256, which is expected since sha512 produces a larger hash output.

3. **NN handshake elapsed time** depicted in Figure 5.84 we can observe the following:
    - The elapsed time for the handshake process varies slightly across different algorithm combinations. Some combinations, like aesgcm-x25519-blake2b and aesgcm-x25519-blake2s, show very similar performance, indicating that changing the hashing algorithm between blake2b and blake2s does not significantly affect the time for the handshake when using aesgcm and x25519.
    - The combinations with x448 consistently show higher elapsed times across different hashing functions, suggesting that using x448 might lead to slower handshakes.
    - The chacha20 encryption with x448 and sha512 exhibits one of the highest elapsed times, which could be indicative of either the computational complexity or the secure nature of this specific combination.

Table 5.28: NN Pattern Handshake - Gateway (Server)

Index	**Handshake Options**	**Memory Usage Bytes**	**Packet Size ClientHello**	**Packet Size ServerHello**	**Packet Size AppData**	**Packet Sent Size Bytes**	**Packet Recv Size Bytes**	**Time Handshake Initiate Sec**	**Time Handshake Process Sec**
1	aesgcm-x2551-sha256	5212	130	146	152	66	61	0	6.896314
2	aesgcm-x2551-sha512	5332	162	178	184	66	61	0	6.897371
3	aesgcm-x2551-blake2s	5198	130	146	152	66	61	0	6.899261
4	aesgcm-x2551-blake2b	5318	162	178	184	66	61	0	6.897831
5	aesgcm-x448d-sha256	5328	178	194	176	90	85	0	7.008675
6	aesgcm-x448d-sha512	5440	210	226	208	90	85	0	6.995463
7	aesgcm-x448d-blake2s	5306	178	194	176	90	85	0	7.007288
8	aesgcm-x448d-blake2b	5418	210	226	208	90	85	0	7.01003
9	chacha-x2551-sha256	5808	126	142	148	66	61	0	6.89812
10	chacha-x2551-sha512	5928	158	174	180	66	61	0	6.89816
11	chacha-x2551-blake2s	5794	126	142	148	66	61	0	6.898098
12	chacha-x2551-blake2b	5914	158	174	180	66	61	0	6.899205
13	chacha-x448d-sha256	5916	174	190	172	90	85	0	7.013565
14	chacha-x448d-sha512	6036	206	222	204	90	85	0	7.011881
15	chacha-x448d-blake2s	5902	174	190	172	90	85	0	7.010919
16	chacha-x448d-blake2b	6014	206	222	204	90	85	0	7.010571

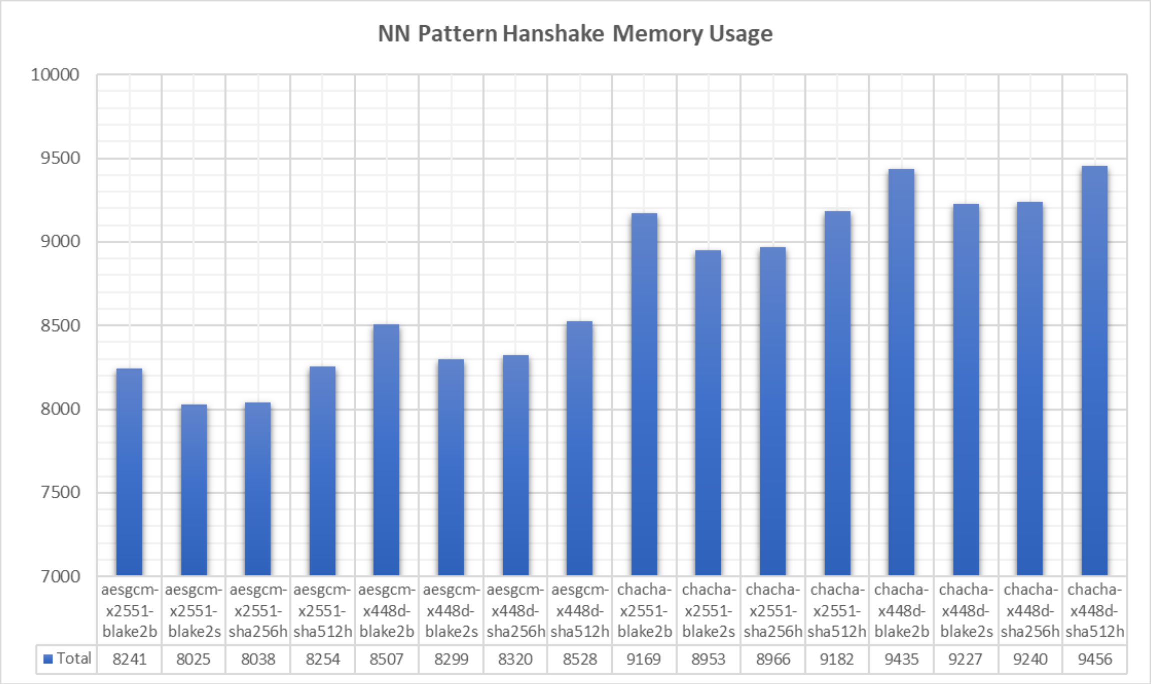


Figure 5.82: NN Handshake Memory Usage associated with different security options

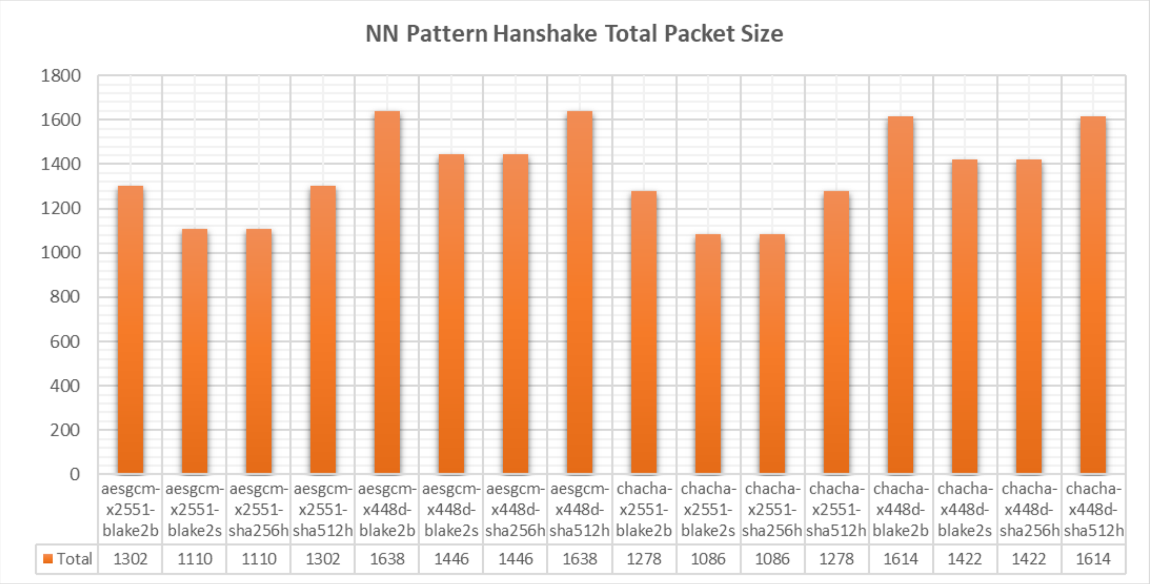


Figure 5.83: NN Handshake Packet Size associated with different security options

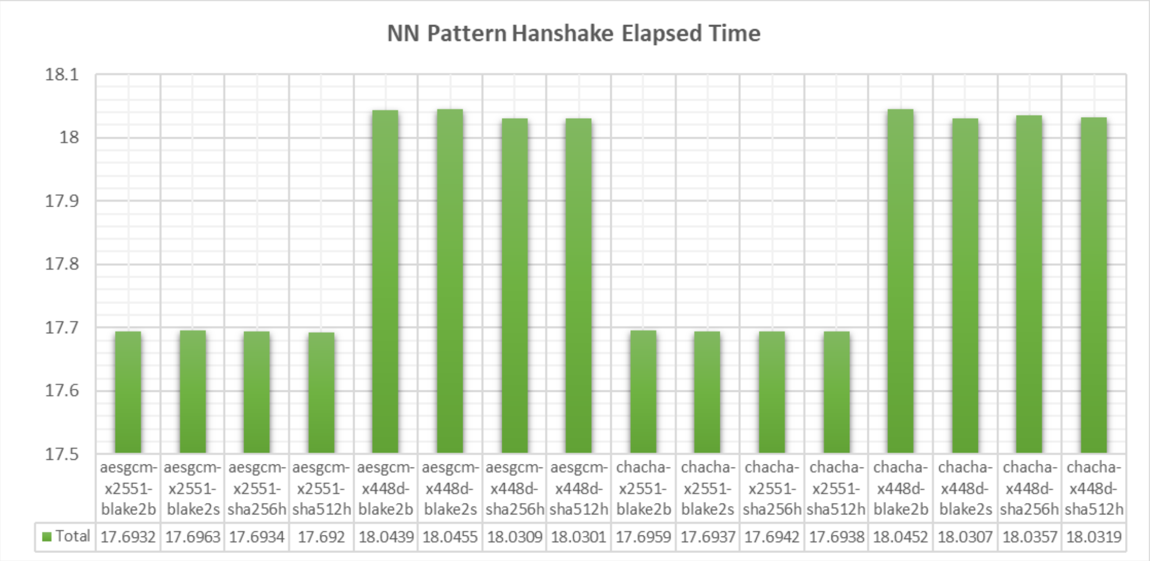


Figure 5.84: NN Handshake Elalapsed Time associated with different security options

NX Pattern

This analysis was presented through two tables 5.29 and 5.30, detailing the process from the perspectives of both the meter and the gateway, and was followed by a chart that summarized the findings. This chart provided a clear overview of memory usage, packet size, and the time required to complete the handshake and transmit meter data.

1. **NX handshake memory usage** depicted in Figure 5.85 we can observe the following:
    - The memory usage across the algorithm combinations is quite uniform, with aesgcm-x2551-blake2b using the least memory and chacha-x2551-sha512 using the most.
    - chacha-x2551 combinations generally consume more memory than aesgcm-x2551 combinations.
2. **NX handshake packet size** depicted in Figure 5.86, we can observe the following:
    - Packet sizes are higher for aesgcm-x2551-blake2b and chacha-x2551-sha512, suggesting these combinations may require more bandwidth.
    - aesgcm-x2551-blake2s and chacha-x2551-blake2s have the smallest packet sizes, indicating they are potentially more efficient in terms of bandwidth usage.
3. **NX handshake elapsed time** depicted in Figure 5.87 we can observe the following:
    - Elapsed time shows minimal variation across all combinations, with most of them completing the handshake in just over 18 seconds.
    - The aesgcm-x2551-sha512 combination has the longest elapsed time, while aesgcm-x2551-blake2b has the shortest, which may suggest slight differences in processing efficiency.

In summary, the NX Pattern indicates that chacha-x2551 combinations tend to use more memory, which could be a consideration for environments with limited memory resources. Packet sizes vary and could impact network throughput, with sha512 hash versions showing larger sizes. Elapsed times are very close, with minor differences that might not have a significant impact on the overall handshake performance.

Table 5.29: NX Pattern Handshake - Meter (Client)

No	Handshake Options	Memory Usage Bytes	Packet Size ClientHello	Packet Size ServerHello	Packet Size AppData	Packet Sent Size Bytes	Packet Recv Size Bytes	Time KeyGen Sec	Time Handshake Initiate Sec	Time Handshake Process Sec
1	aesgcm-x2551-sha256	2826	130	194	152	50	125	0.007713	0.000997	11.02571
2	aesgcm-x2551-sha512	2922	162	226	184	50	125	0	0	11.0228
3	aesgcm-x2551-blake2s	2827	130	194	152	50	125	0	0	11.01904
4	aesgcm-x2551-blake2b	2923	162	226	184	50	125	0	0	11.03546
5	chacha-x2551-sha256	3158	126	190	148	50	125	0.006648	0.000908	11.02884
6	chacha-x2551-sha512	3254	158	222	180	50	125	0	0	11.0359
7	chacha-x2551-blake2s	3159	126	190	148	50	125	0	0	11.03567
8	chacha-x2551-blake2b	3255	158	222	180	50	125	0	0	11.03556

Table 5.30: NX Pattern Handshake - Gateway (Server)

No	Handshake Options	Memory Usage Bytes	Packet Size ClientHello	Packet Size ServerHello	Packet Size AppData	Packet Sent Size Bytes	Packet Recv Size Bytes	Time Handshake Initiate Sec	Time Handshake Process Sec
1	aesgcm-x2551-sha256h	5260	130	194	152	114	61	0	7.132645
2	aesgcm-x2551-sha512h	5380	162	226	184	114	61	0	7.125195
3	aesgcm-x2551-blake2s	5246	130	194	152	114	61	0.015625	7.121336
4	aesgcm-x2551-blake2b	5366	162	226	184	114	61	0	7.137256
5	chacha-x2551-sha256h	6289	126	190	148	114	61	0	7.136366
6	chacha-x2551-sha512h	6409	158	222	180	114	61	0	7.133103
7	chacha-x2551-blake2s	6275	126	190	148	114	61	0	7.138229
8	chacha-x2551-blake2b	6387	158	222	180	114	61	0	7.138067

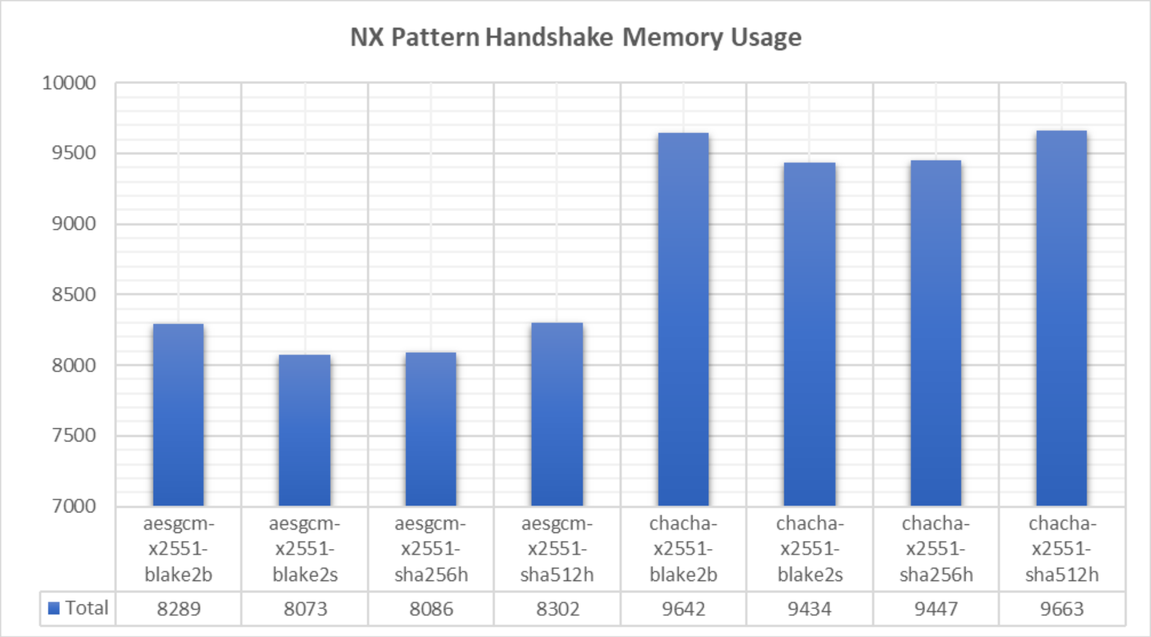


Figure 5.85: NX Handshake Memory Usage associated with different security options

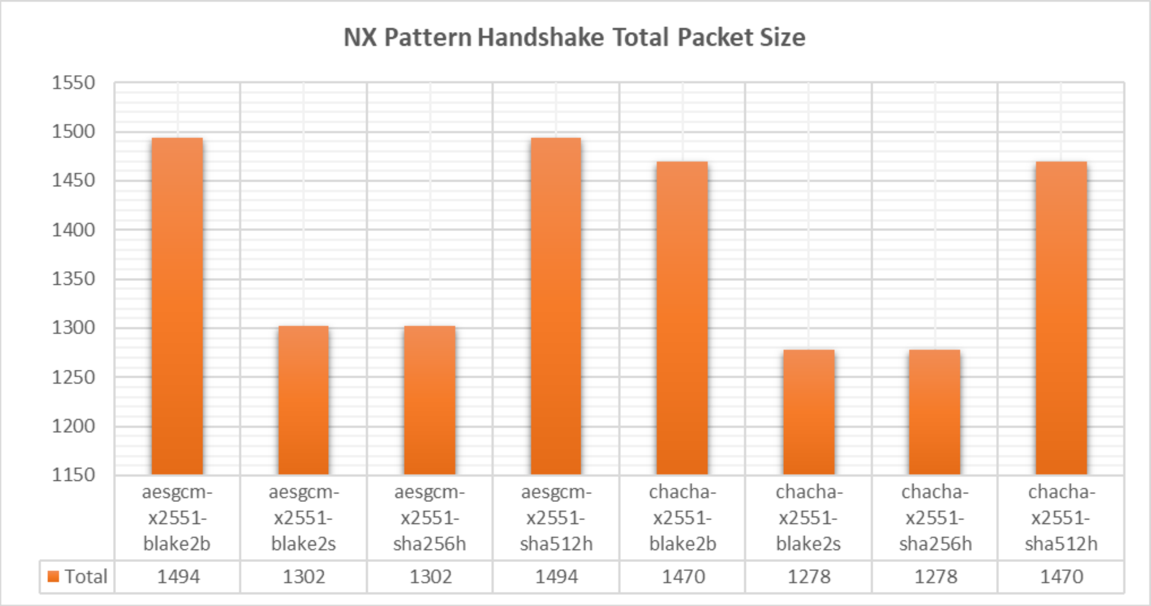


Figure 5.86: NX Handshake Packet Size associated with different security options

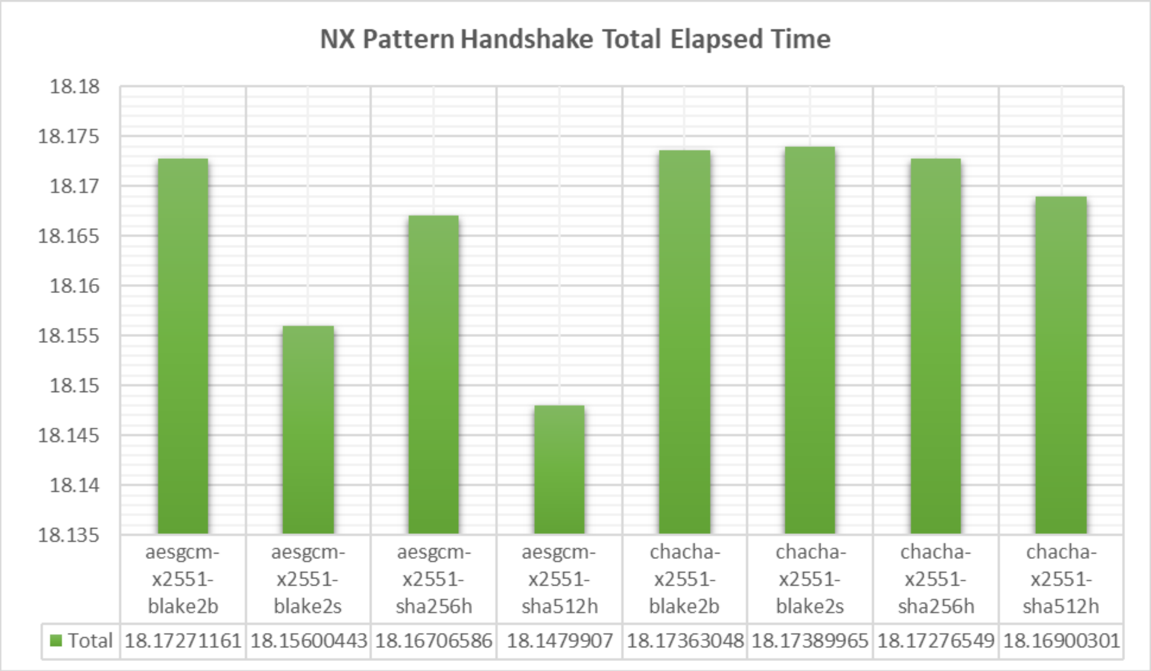


Figure 5.87: NX Handshake Packet Size associated with different security options

5.6.2 NPF - Three Messages

XK Pattern

This analysis was presented through two tables 5.31 and 5.32, detailing the process from the perspectives of both the meter and the gateway, and was followed by a chart that summarized the findings. This chart provided a clear overview of memory usage, packet size, and the time required to complete the handshake and transmit meter data.

1. **XK handshake memory usage** depicted in Figure 5.88 we can observe the following:
   - Memory usage among the cryptographic combinations varies significantly, with aesgcm-x2551-sha512 consuming by far the most memory.
   - aesgcm-x448 combinations exhibit lower memory usage compared to aesgcm-x2551 combinations, with the exception of aesgcm-x448-sha512, which still uses less memory than its x2551 counterpart.
   - chacha-x448 combinations generally consume more memory than aesgcm-x448, but less than aesgcm-x2551-sha512.
2. **XK handshake packet size** depicted in Figure 5.89, we can observe the following:
   - The packet sizes are greater for aesgcm-x448 combinations, particularly for those using sha512, indicating a potential increase in bandwidth usage.
   - aesgcm-x2551-blake2b and chacha-x2551 combinations have smaller packet sizes, which could indicate better efficiency in terms of bandwidth usage.
3. **XK handshake elapsed time** depicted in Figure 5.90 we can observe the following:
   - The elapsed time shows minimal variation, with most combinations completing the handshake process in the 22.5 to 22.9 seconds range.
   - There's a notable increase in time for aesgcm-x448-sha512 and chacha-x448-sha512, which may suggest that these combinations are slightly less efficient in terms of processing speed.

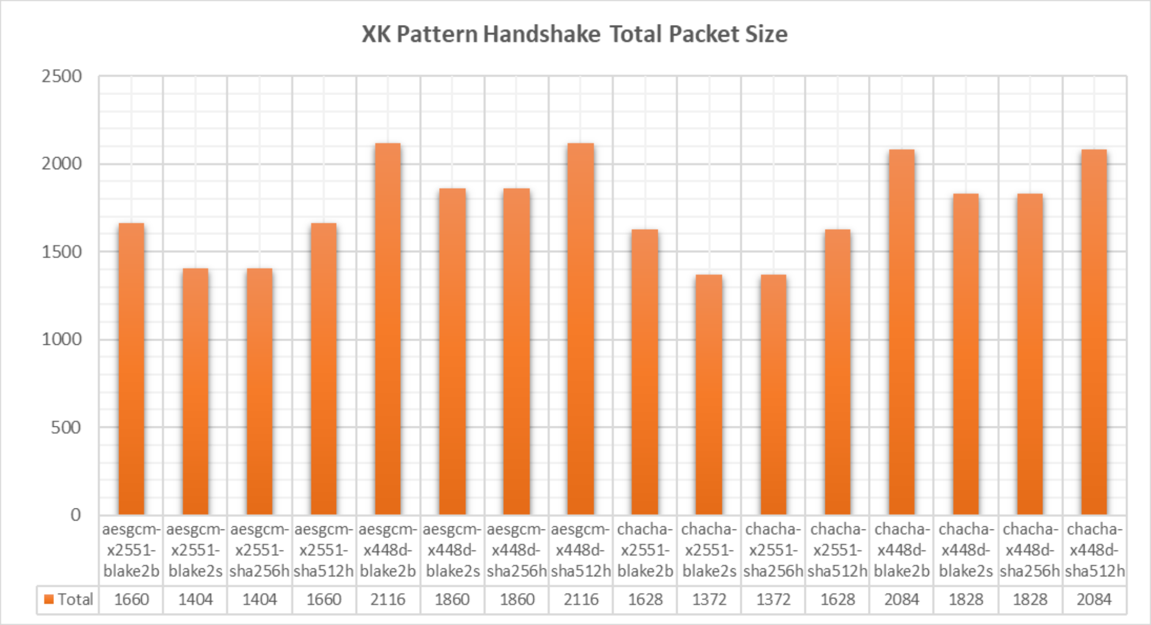
In summary, the XK Pattern indicates that the choice of cryptographic algorithm can have a significant impact on memory usage, with aesgcm-x2551-sha512 standing out as particularly resource-intensive. Packet sizes and elapsed times also vary, but to a lesser extent, suggesting that the overall impact on handshake duration might be less pronounced.

Table 5.31: XK Pattern Handshake - Meter (Client)

No	Handshake Options	Memory Usage Bytes	Packet Size ClientHello	Packet Size ServerHello	Packet Size ClientFinish	Packet Size AppData	Packet Sent Size Bytes	Packet Recv Size Bytes	Time Handshake Initiate Sec	Time Handshake Process Sec
1	aesgcm-x2551-sha256	18622	130	130	146	152	116	61	0.002998	11.72478
2	aesgcm-x2551-sha512	10671	162	162	178	184	116	61	0.000998	11.71679
3	aesgcm-x2551-blake2s	9112	130	130	146	152	116	61	0.001998	11.72681
4	aesgcm-x2551-blake2b	8680	162	162	178	184	116	61	0.000999	11.7143
5	aesgcm-x448d-sha256	8360	178	178	194	176	164	85	0.000999	11.95814
6	aesgcm-x448d-sha512	8168	210	210	226	208	164	85	0.000999	11.95663
7	aesgcm-x448d-blake2s	7481	178	178	194	176	164	85	0.000999	11.95464
8	aesgcm-x448d-blake2b	7265	210	210	226	208	164	85	0.001001	11.95863
9	chacha-x2551-sha256	10767	126	126	142	148	116	61	0.001998	11.71578
10	chacha-x2551-sha512	8854	158	158	174	180	116	61	0.001	11.71778
11	chacha-x2551-blake2s	8151	126	126	142	148	116	61	0.002002	11.71512
12	chacha-x2551-blake2b	8015	158	158	174	180	116	61	0.001004	11.71678
13	chacha-x448d-sha256	8276	174	174	190	172	164	85	0.001	11.94605
14	chacha-x448d-sha512	8076	206	206	222	204	164	85	0.000999	11.95463
15	chacha-x448d-blake2s	7581	174	174	190	172	164	85	0.001002	11.96463
16	chacha-x448d-blake2b	7637	206	206	222	204	164	85	0.000998	11.95063

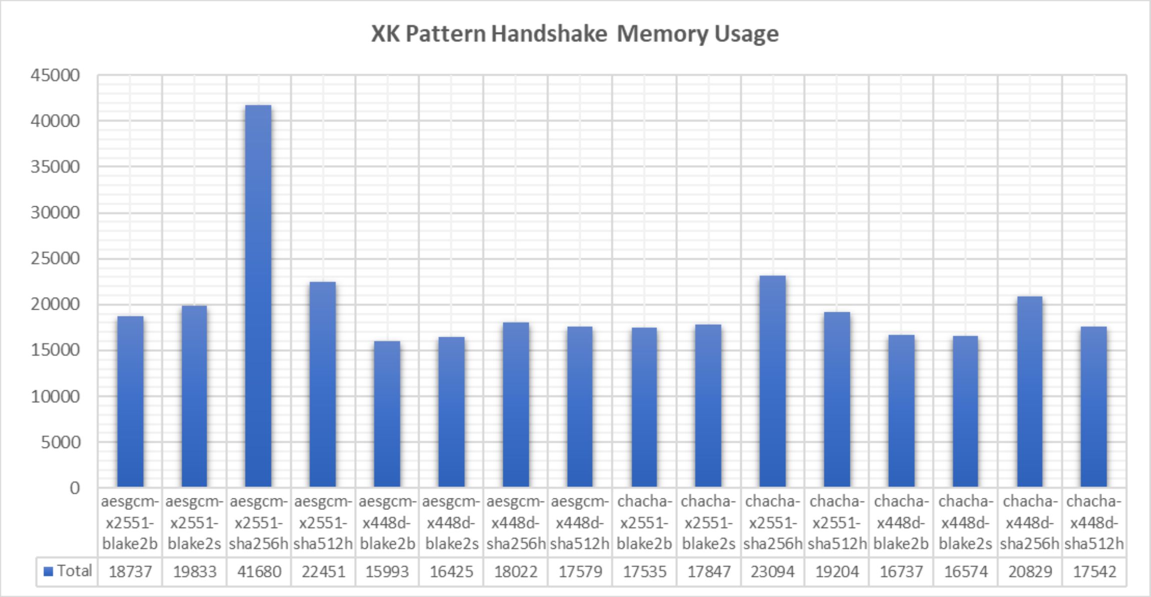
Table 5.32: XK Pattern Handshake - Gateway (Server)

No	Handshake Options	Memory Usage Bytes	Packet Size ClientHello	Packet Size ServerHello	Packet Size ClientFinish	Packet Size AppData	Packet Calculated Size Bytes	Packet Sent Size Bytes	Packet Recv Size Bytes	Time Handshake Initiate Sec	Time Handshake Process Sec
1	aesgcm-x2551-sha256	23058	130	130	146	152	130	50	61	0.003995	10.82033
2	aesgcm-x2551-sha512	11780	162	162	178	184	162	50	61	0.002996	10.82334
3	aesgcm-x2551-blake2s	10721	130	130	146	152	130	50	61	0.004997	10.82136
4	aesgcm-x2551-blake2b	10057	162	162	178	184	162	50	61	0.002997	10.80985
5	aesgcm-x448d-sha256	9662	178	178	194	176	178	74	85	0.004	10.93577
6	aesgcm-x448d-sha512	9411	210	210	226	208	210	74	85	0.003998	10.93626
7	aesgcm-x448d-blake2s	8944	178	178	194	176	178	74	85	0.004993	10.93227
8	aesgcm-x448d-blake2b	8728	210	210	226	208	210	74	85	0.003995	10.95025
9	chacha-x2551-sha256	12327	126	126	142	148	126	50	61	0.002998	10.80934
10	chacha-x2551-sha512	10350	158	158	174	180	158	50	61	0.003	10.82333
11	chacha-x2551-blake2s	9696	126	126	142	148	126	50	61	0.004993	10.81867
12	chacha-x2551-blake2b	9520	158	158	174	180	158	50	61	0.002997	10.82433
13	chacha-x448d-sha256	12553	174	174	190	172	174	74	85	0.003998	10.93368
14	chacha-x448d-sha512	9466	206	206	222	204	206	74	85	0.003999	10.93526
15	chacha-x448d-blake2s	8993	174	174	190	172	174	74	85	0.003996	10.93726
16	chacha-x448d-blake2b	9100	206	206	222	204	206	74	85	0.003998	10.93526



	aesgcm-x2551-blake2b	aesgcm-x2551-blake2s	aesgcm-x2551-sha256h	aesgcm-x2551-sha512h	aesgcm-x448d-blake2b	aesgcm-x448d-blake2s	aesgcm-x448d-sha256h	aesgcm-x448d-sha512h	chacha-x2551-blake2b	chacha-x2551-blake2s	chacha-x2551-sha256h	chacha-x2551-sha512h	chacha-x448d-blake2b	chacha-x448d-blake2s	chacha-x448d-sha256h	chacha-x448d-sha512h
■ Total	1660	1404	1404	1660	2116	1860	1860	2116	1628	1372	1372	1628	2084	1828	1828	2084

Figure 5.89: Xk Handshake Packet Size associated with different security options



	aesgcm-x2551-blake2b	aesgcm-x2551-blake2s	aesgcm-x2551-sha256h	aesgcm-x2551-sha512h	aesgcm-x448d-blake2b	aesgcm-x448d-blake2s	aesgcm-x448d-sha256h	aesgcm-x448d-sha512h	chacha-x2551-blake2b	chacha-x2551-blake2s	chacha-x2551-sha256h	chacha-x2551-sha512h	chacha-x448d-blake2b	chacha-x448d-blake2s	chacha-x448d-sha256h	chacha-x448d-sha512h
■ Total	18737	19833	41680	22451	15993	16425	18022	17579	17535	17847	23094	19204	16737	16574	20829	17542

Figure 5.88: XK Handshake Memory Usage associated with different security options

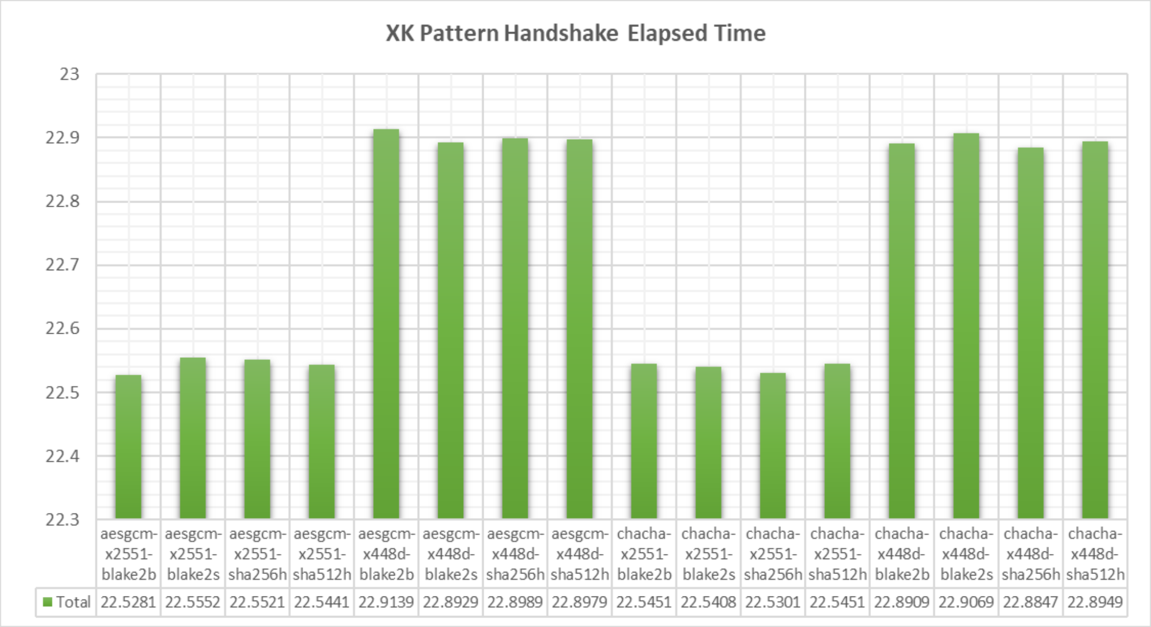


Figure 5.90: XK Handshake Packet Size associated with different security options

XN Pattern

This analysis was presented through two tables 5.33 and 5.34, detailing the process from the perspectives of both the meter and the gateway, and was followed by a chart that summarized the findings. This chart provided a clear overview of memory usage, packet size, and the time required to complete the handshake and transmit meter data.

1. **XN handshake memory usage** depicted in Figure 5.91 we can observe the following:
    - There is a moderate variation in memory usage across the cryptographic combinations.
    - The chacha-x448-sha512 combination uses the most memory, significantly more than the others.
    - aesgcm-x2551-blake2b shows the least memory usage among the combinations, with the rest fairly distributed in between.
2. **XN handshake packet size** depicted in Figure 5.92, we can observe the following:
    - The packet sizes vary with aesgcm-x448-blake2b and chacha-x448-sha512 having the largest packet sizes, which may affect bandwidth requirements.

Table 5.33: XN Pattern Handshake - Meter (Client)

No	Handshake Options	Memory Usage Bytes	Packet Size ClientHello	Packet Size ServerHello	Packet Size ClientFinish	Packet Size AppData	Packet Sent Size Bytes	Packet Recv Size Bytes	Time Handshake Initiate Sec	Time Handshake Process Sec
1	aesgcm-x2551-sha256h	5288	114	130	146	152	100	61	0.000999	11.63886
2	aesgcm-x2551-sha512h	5456	146	162	178	184	100	61	0	11.63983
3	aesgcm-x2551-blake2s	5193	114	130	146	152	100	61	0.001998	11.63592
4	aesgcm-x2551-blake2b	5377	146	162	178	184	100	61	0.002	11.63834
5	aesgcm-x448d-sha256h	5382	162	178	194	176	148	85	0.000999	11.8618
6	aesgcm-x448d-sha512h	5558	194	210	226	208	148	85	0.000999	11.87669
7	aesgcm-x448d-blake2s	5295	162	178	194	176	148	85	0.000998	11.87368
8	aesgcm-x448d-blake2b	5439	194	210	226	208	148	85	0	11.88276
9	chacha-x2551-sha256h	6228	110	126	142	148	100	61	0.000999	11.63583
10	chacha-x2551-sha512h	6420	142	158	174	180	100	61	0.000997	11.63683
11	chacha-x2551-blake2s	6173	110	126	142	148	100	61	0.000999	11.63583
12	chacha-x2551-blake2b	6381	142	158	174	180	100	61	0.000999	11.63783
13	chacha-x448d-sha256h	9916	158	174	190	172	148	85	0.001	11.88225
14	chacha-x448d-sha512h	6626	190	206	222	204	148	85	0.000999	11.8736
15	chacha-x448d-blake2s	6395	158	174	190	172	148	85	0	11.87819
16	chacha-x448d-blake2b	6627	190	206	222	204	148	85	0.000999	11.87642

- Other combinations such as aesgcm-x2551-blake2s and chacha-x2551-blake2s exhibit smaller packet sizes, suggesting better bandwidth efficiency.

3. **XN handshake elapsed time** depicted in Figure 5.93 we can observe the following:

    - The elapsed time for the handshake process is fairly consistent across all combinations, with most completing the handshake in around 22.4 to 22.9 seconds.
    - There's no significant outlier in elapsed time, indicating that processing speed is relatively uniform across these cryptographic protocols.

In summary, the XN Pattern demonstrates that while there are variations in memory and packet size among the different cryptographic algorithm combinations, the overall impact on handshake duration is minimal. The 'chacha-x448-sha512' combination stands out for higher resource usage, which could be a consideration for systems with memory constraints or lower bandwidth availability. The findings from these graphs could inform the selection of cryptographic protocols in various applications, balancing between security strength and resource efficiency.

XX Pattern

This analysis was presented through two tables 5.35 and 5.36, detailing the process from the perspectives of both the meter and the gateway, and was followed by a chart that summarized the findings. This chart provided a clear overview of memory usage, packet size, and the time required to complete the handshake and transmit meter data.

1. **XX handshake memory usage** depicted in Figure 5.94 we can observe the following:

Table 5.34: XN Pattern Handshake - Gateway (Server)

Index	Handshake Options	Memory Usage Bytes	Packet Size ClientHello	Packet Size ServerHello	Packet Size ClientFinish	Packet Size AppData	Packet Sent Size Bytes	Packet Recv Size Bytes	Time Handshake Initiate Sec	Time Handshake Process Sec
1	aesgcm-x2551-sha256h	6953	114	130	146	152	50	45	0.002996	10.82336
2	aesgcm-x2551-sha512h	7057	146	162	178	184	50	45	0.002	10.82433
3	aesgcm-x2551-blake2s	6779	114	130	146	152	50	45	0.002	10.82441
4	aesgcm-x2551-blake2b	6883	146	162	178	184	50	45	0.002999	10.82384
5	aesgcm-x448d-sha256h	7005	162	178	194	176	74	69	0.001997	10.93337
6	aesgcm-x448d-sha512h	7117	194	210	226	208	74	69	0.001999	10.93727
7	aesgcm-x448d-blake2s	6839	162	178	194	176	74	69	0.001999	10.93826
8	aesgcm-x448d-blake2b	6903	194	210	226	208	74	69	0.001998	10.95233
9	chacha-x2551-sha256h	7693	110	126	142	148	50	45	0.001999	10.81134
10	chacha-x2551-sha512h	7845	142	158	174	180	50	45	0.001996	10.82333
11	chacha-x2551-blake2s	7631	110	126	142	148	50	45	0.002001	10.82433
12	chacha-x2551-blake2b	7807	142	158	174	180	50	45	0.001999	10.82633
13	chacha-x448d-sha256h	7873	158	174	190	172	74	69	0.001998	10.94179
14	chacha-x448d-sha512h	8049	190	206	222	204	74	69	0.002998	10.93728
15	chacha-x448d-blake2s	7851	158	174	190	172	74	69	0.001996	10.93777
16	chacha-x448d-blake2b	8027	190	206	222	204	74	69	0.001999	10.938

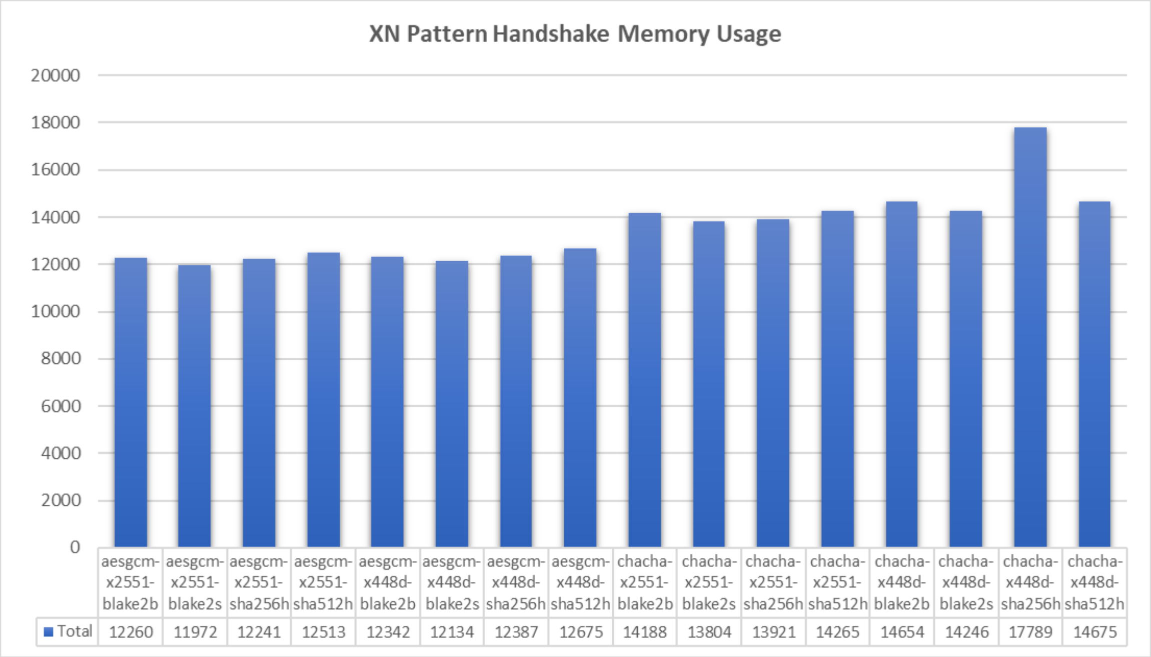


Figure 5.91: XN Handshake Memory Usage associated with different security options

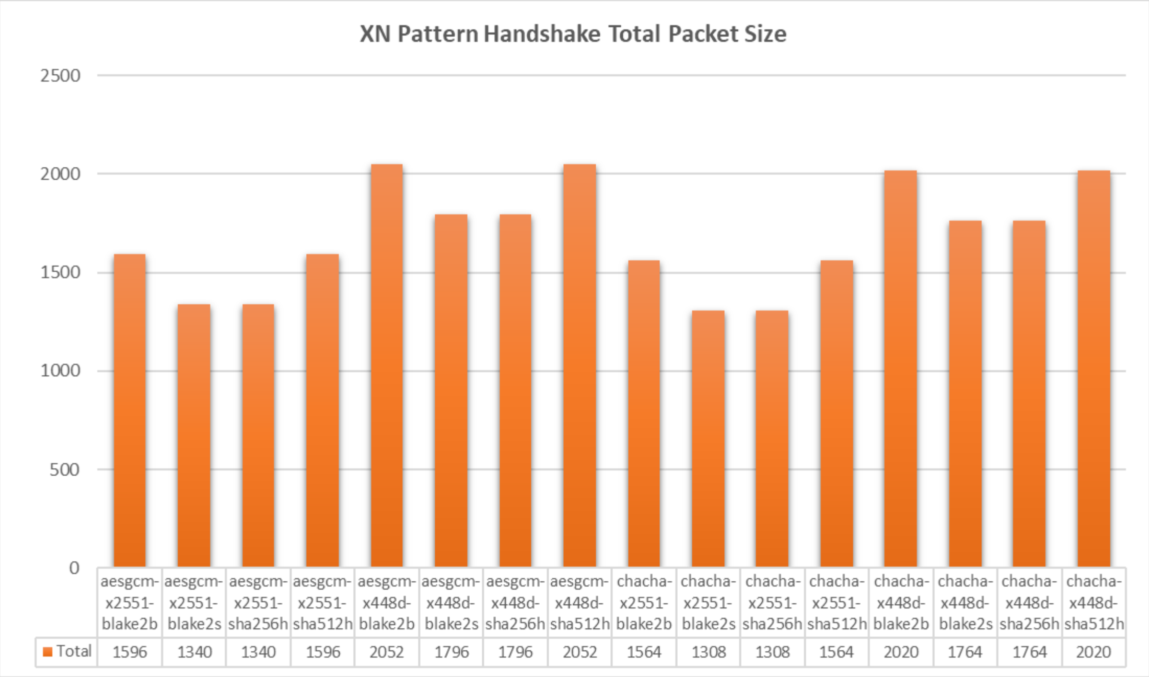


Figure 5.92: XN Handshake Packet Size associated with different security options

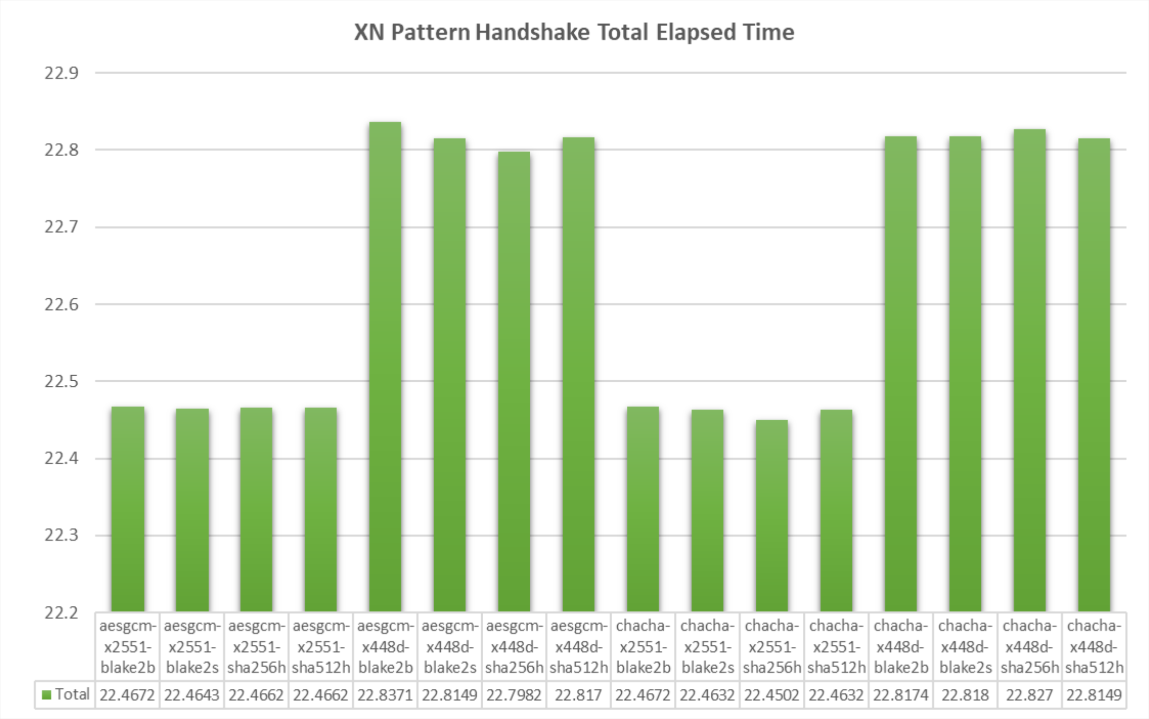


Figure 5.93: XN Handshake Elapsed Time associated with different security options

Table 5.35: XX Pattern Handshake - Meter (Client)

Index	**Handshake Options**	**Memory Usage Bytes**	**Packet Size ClientHello**	**Packet Size ServerHello**	**Packet Size ClientFinish**	**Packet Size AppData**	**Packet Sent Size Bytes**	**Packet Recv Size Bytes**	**Time Handshake Initiate Sec**	**Time Handshake Process Sec**
1	aesgcm-x2551-sha256h	5177	114	178	146	152	100	109	0.001	11.87468
2	aesgcm-x2551-sha512h	5393	146	210	178	184	100	109	0.001001	11.87668
3	aesgcm-x2551-blake2s	5162	114	178	146	152	100	109	0.000999	11.87668
4	aesgcm-x2551-blake2b	5378	146	210	178	184	100	109	0.000999	11.87927
5	chacha-x2551-sha256h	6854	110	174	142	148	100	109	0.002997	11.86769
6	chacha-x2551-sha512h	7070	142	206	174	180	100	109	0.000999	11.87769
7	chacha-x2551-blake2s	6839	110	174	142	148	100	109	0	11.88268
8	chacha-x2551-blake2b	7047	142	206	174	180	100	109	0.000999	11.87521

- aesgcm-x2551-blake2b uses the least amount of memory, while chacha-x2551-sha512 uses the most.
- The chacha-x2551 combinations have a higher memory usage compared to aesgcm-x2551 combinations, with chacha-x2551-sha512 being the most memory-intensive.

2. **XX handshake packet size** depicted in Figure 5.95, we can observe the following:
    - The packet size is largest for aesgcm-x2551-sha512 and smallest for aesgcm-x2551-blake2s.
    - chacha-x2551 combinations result in larger packet sizes compared to aesgcm-x2551, indicating a higher bandwidth requirement for Chacha based combinations.
3. **XX handshake elapsed time** depicted in Figure 5.96 we can observe the following:
    - The handshake completion time is consistent across all combinations, with a slight increase in time for chacha-x2551-sha512.
    - The aesgcm-x2551 combinations have similar elapsed times, indicating comparable efficiency.

In conclusion, the XX Pattern demonstrates that chacha-x2551-sha512 tends to use more resources in terms of memory and packet size, which could impact systems with limited resources. The handshake times are relatively consistent across the board, suggesting that the difference in processing speed is minimal among the tested cryptographic combinations. These observations are critical for system architects and security engineers when choosing suitable cryptographic protocols for their systems, considering the trade-off between security strength and resource consumption.

Table 5.36: XX Pattern Handshake - Gateway (Server)

No	Handshake Options	Memory Usage Bytes	Packet Size ClientHello	Packet Size ServerHello	Packet Size ClientFinish	Packet Size AppData	Packet Sent Size Bytes	Packet Recv Size Bytes	Time Handshake Initiate Sec	Time Handshake Process Sec
1	aesgcm-x2551-sha256h	6433	114	178	146	152	98	45	0.001999	11.05119
2	aesgcm-x2551-sha512h	6617	146	210	178	184	98	45	0.001998	11.05319
3	aesgcm-x2551-blake2s	6419	114	178	146	152	98	45	0.001999	11.05119
4	aesgcm-x2551-blake2b	6603	146	210	178	184	98	45	0.001999	11.0522
5	chacha-x2551-sha256h	8118	110	174	142	148	98	45	0.001998	11.04919
6	chacha-x2551-sha512h	8302	142	206	174	180	98	45	0.003000	11.05119
7	chacha-x2551-blake2s	8104	110	174	142	148	98	45	0.001999	11.05319
8	chacha-x2551-blake2b	8280	142	206	174	180	98	45	0.001998	11.05072

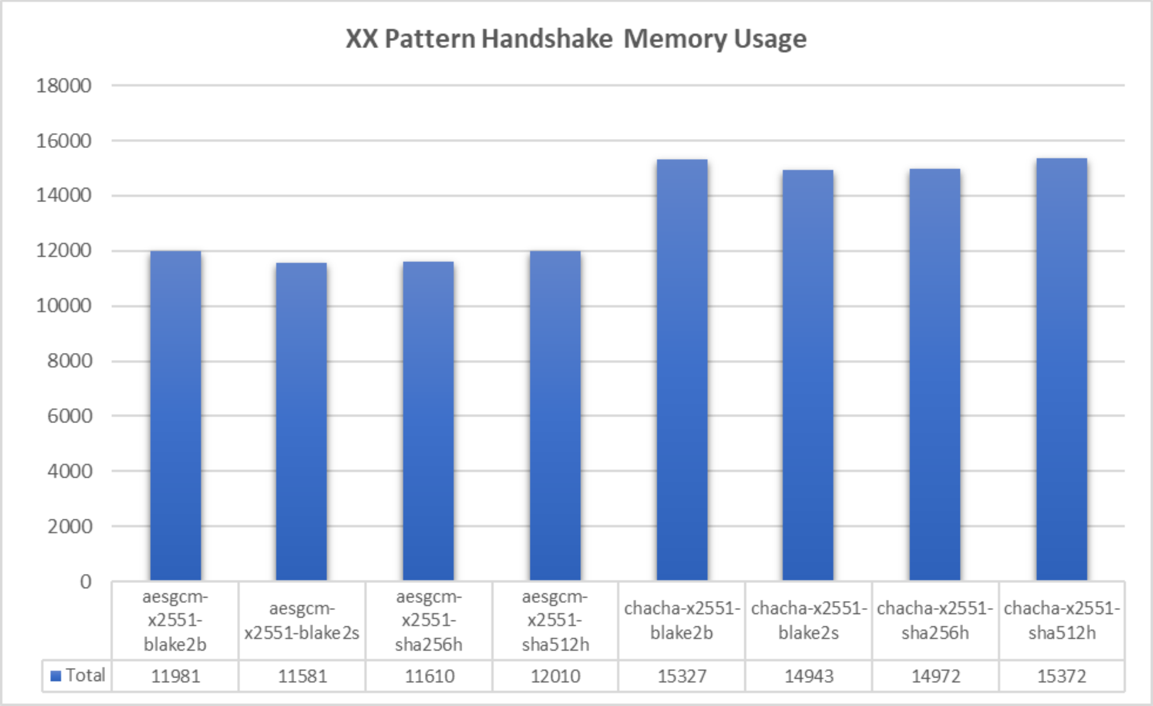


Figure 5.94: XX Handshake Memory Usage associated with different security options

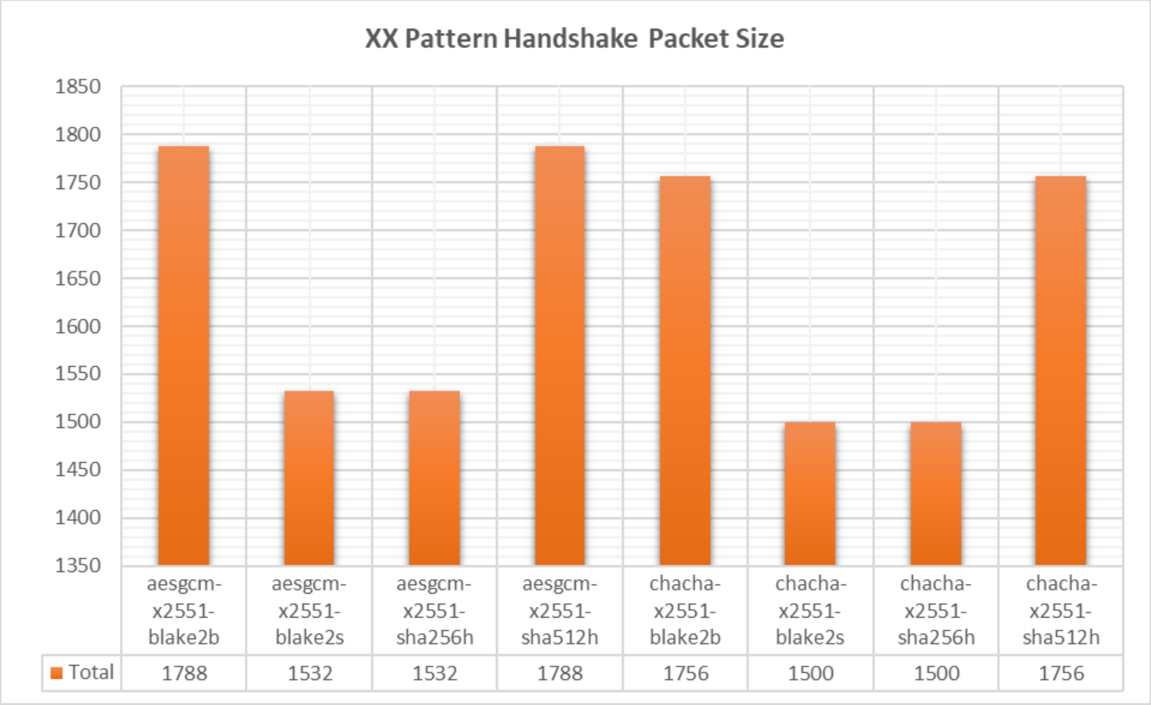


Figure 5.95: XX Handshake Packet Size associated with different security options

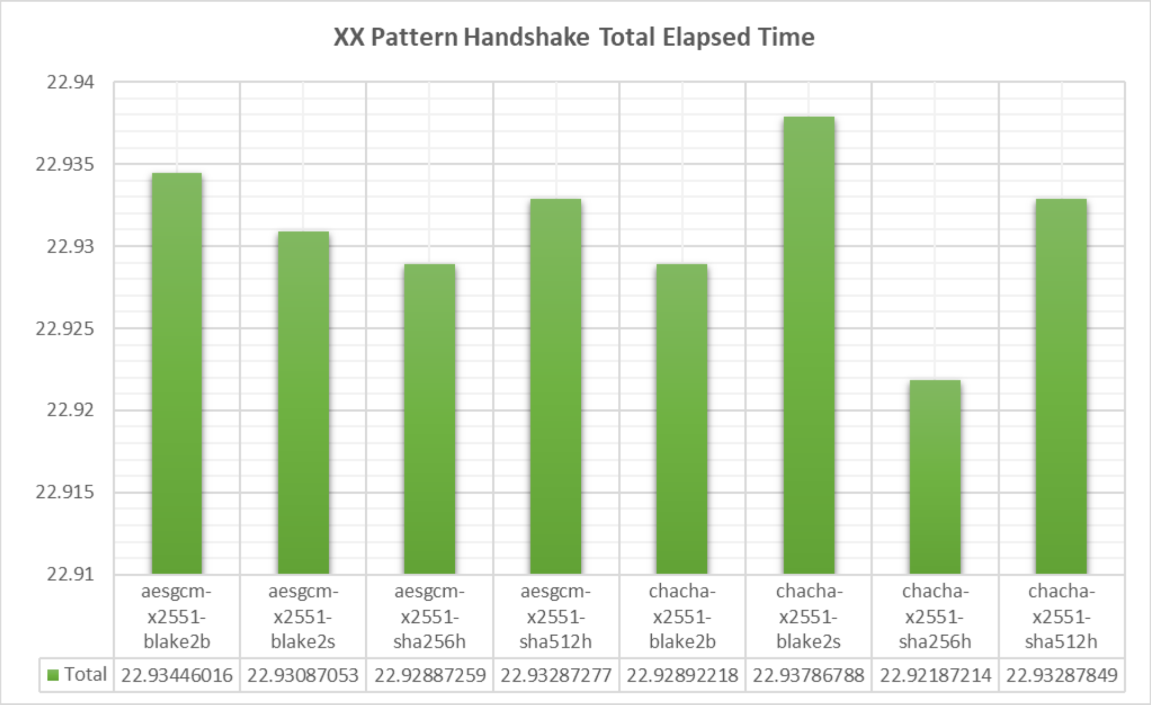


Figure 5.96: XX Handshake Elapsed Time associated with different security options

Chapter 6

6 Evaluation and Analysis

This chapter provides a detailed evaluation of the implementation outcomes for the NPF-based security and privacy protocol over wM-Bus networks. The chapter is structured around four main analyses that collectively assess the protocol's performance, security, and efficiency.

The first section presents a comparative analysis of all the predefined patterns within the Noise framework. This analysis serves as a foundation for understanding the relative advantages and disadvantages of each pattern in the context of wM-Bus communication.

Next, we compare the Noise framework, specifically focusing on the NX and XX patterns, with TLS. This comparison is essential to highlight the benefits and potential limitations of using the Noise framework instead of TLS for securing IoT ecosystems.

In the third part, we introduce a detailed threat model to evaluate the security of the proposed protocol against various attacks. This analysis uses the STRIDE methodology to ensure that the protocol can withstand common security threats and provide robust protection for wM-Bus communications.

Finally, the chapter concludes with an examination of the protocol's lightweight nature by analyzing the battery life implications for the NX, XX patterns, and TLS. This analysis demonstrates the protocol's suitability for energy-constrained IoT devices, emphasizing its minimal impact on battery life while maintaining high-security standards.

6.1 Nosie Protocol Framework Patterns Comparative Analysis

1. Stage 1 comparing the first 3 patterns IK, IN, and IX:
    - Based on this analysis, Pattern IX appears to be the most suitable option as it has low memory usage, moderate packet size, and a shorter elapsed time for the handshake process. However, we can wait for the analysis of the remaining patterns before making a final decision. Let me know if you'd like to proceed with the next set of patterns or if you have any specific preferences.
2. Stage 2 comparing the second 3 patterns KK, KN, KX, with the previous winning pattern IX:

  - Pattern KX has similar characteristics to pattern IX, with low memory usage, moderate packet size, and a shorter elapsed time for the handshake process.
  - Patterns KK and KN have similar characteristics to patterns IK and IN, respectively.
  - Based on this comparison, Pattern KX appears to be the winning pattern, similar to pattern IX. It offers a balance of low memory usage, moderate packet size, and a shorter handshake time.

3. Stage 3 Comparing NK, NN, and NX patterns with the previous winning pattern KX:
   - Pattern NX has similar characteristics to pattern KX, with low memory usage, small packet size, and a shorter elapsed time for the handshake process.
   - Patterns NK and NN have characteristics similar to previous patterns IK and IN, respectively.
   - Based on this comparison, Pattern NX appears to be the winning pattern, similar to pattern KX. It offers low memory usage, small packet size, and a shorter handshake time.
4. Stage 4: Comparing XK, XN, and XX patterns with the previous winning pattern NX:
   - Pattern XX has similar characteristics to pattern NX, with low memory usage, small packet size, and a shorter elapsed time for the handshake process.
   - Patterns XK and XN have characteristics similar to previous patterns IK and IN, respectively.
   - Based on this comparison, Pattern XX appears to be the winning pattern, similar to pattern NX. It offers low memory usage, small packet size, and a shorter handshake time.

Consequently, after thorough evaluation and analysis, it has been conclusively determined that Pattern XX shall be adopted for the implementation of the three-message handshake protocol. Conversely, Pattern NX has been identified as the optimal choice for the two-message handshake mechanism. These selections are predicated on their superior compatibility with the wM-Bus framework, particularly in terms of offering robust security features without compromising on the system's need for lightweight and efficient operation.. An overview comparison depicted in Table 6.1.

Table 6.1: Comparison of all Noise Framework Patterns

Pattern	**Memory Usage**	**Packet Size**	**Handshake Time**
IK	Moderate	Small	Moderate
IN	High	Large	Long
IX	Low	Moderate	Short
KK	Moderate	Small	Moderate
KN	High	Large	Long
KX	Low	Moderate	Short
NK	Moderate	Moderate	Moderate
NN	High	Large	Long
NX	**Low**	**Small**	**Short**
XK	Moderate	Moderate	Moderate
XN	High	Large	Long
XX	**Low**	**Small**	**Short**

Looking into the security options for each pattern XX and NX in terms of best security options such as Cipher, Diffie-Hellman, and Hashing algorithms:

1. **NX Pattern**
   (a) **Cipher Options:**
      - ChaCha20-Poly1305: This cipher is an excellent choice for environments where AES hardware acceleration is not available. ChaCha20-Poly1305 is known for its high performance in software implementations, making it particularly suitable for devices with limited processing power. Its security level is comparable to AES-GCM, but it can often outperform AES-GCM on platforms without AES-NI (a set of instructions for hardware acceleration of AES operations).

   (b) **Diffie-Hellman Options:**
      - x25519: For the DH mechanism, x25519 remains the optimal choice due

to its balance of security and performance. It is an elliptic curve Diffie-Hellman key exchange algorithm that provides strong security with relatively low computational requirements, which is ideal for the lightweight nature of the NX pattern in wM-Bus applications.

(c) **Hashing Options:**

- BLAKE2s: Given the focus on efficiency and lightweight operations for the NX pattern, BLAKE2s could be more suitable than SHA-256 in this scenario. BLAKE2s is optimized for 8- to 32-bit platforms and is designed to be fast and secure on a wide range of devices. It provides cryptographic security comparable to or better than SHA-256 but is faster in software implementations, especially on devices that lack specialized cryptographic hardware.

(d) Proposed NX Pattern Configuration:

- Cipher: ChaCha20-Poly1305, for its high performance in software and strong security, making it well-suited for devices without AES hardware acceleration.
- Diffie-Hellman: x25519, for its efficient and secure key exchange capabilities, ensuring robust security with minimal computational overhead.
- Hashing: BLAKE2s, for its speed and efficiency on a variety of platforms, providing strong security while being more lightweight than SHA-256.

This configuration for the NX pattern in wM-Bus communications offers a balanced approach, ensuring high security and efficient operation even on resource-constrained devices. ChaCha20-Poly1305 and BLAKE2s are particularly well-suited for environments where computational resources are limited, and their selection, along with x25519, aligns with the objectives of securing wM-Bus networks effectively while maintaining optimal performance and resource efficiency.

2. **XX Pattern**

(a) **Cipher Options:**

- The cipher options include AES-GCM (Advanced Encryption Standard Galois/Counter Mode) and ChaCha20-Poly1305 with different key exchange algorithms (x25519) and hashing algorithms (SHA-256, SHA-512, Blake2s, Blake2b).

  - AES-GCM and ChaCha20-Poly1305 are both widely used symmetric encryption algorithms known for their strong security and efficiency.
  - The choice between AES-GCM and ChaCha20-Poly1305 depends on factors such as hardware support, performance requirements, and compatibility with existing systems.

(b) **Diffie-Hellman Options:**

  - The Diffie-Hellman options include x25519, which is an elliptic curve Diffie-Hellman (ECDH) key exchange algorithm.
  - x25519 is a modern and efficient key exchange algorithm based on elliptic curve cryptography, offering strong security with relatively low computational overhead.

(c) **Hashing Options:**

  - The hashing options include SHA-256, SHA-512, Blake2s, and Blake2b.
  - SHA-256 and SHA-512 are well-known cryptographic hash functions widely used for data integrity verification and authentication.
  - Blake2s and Blake2b are high-speed cryptographic hash functions designed for efficient hashing with strong security properties.

(d) Based on these options, the most suitable set of security options for pattern XX could include:

  - Cipher: ChaCha20-Poly1305 with x25519 key exchange
  - Diffie-Hellman: x25519
  - Hashing: SHA-256

These options offer a balance between strong security, efficiency, and lightweight characteristics, making them ideal for securing wM-Bus networks with the XX pattern. ChaCha20-Poly1305 is known for its efficiency and security, x25519 provides strong key exchange capabilities with low computational overhead, and SHA-256 ensures data integrity and authentication without significantly increasing memory usage. By selecting this set of security options, pattern

XX can effectively secure wM-Bus networks while maintaining optimal performance and resource efficiency.

6.2 Comparison Between TLS and NPF NX and XX

The Wireless Meter-Bus (M-Bus) is a European standard (EN 13757-4) designed for the remote reading of gas or electricity meters. With the growing adoption of smart meters and the Internet of Things (IoT), the security of communications in these devices has become paramount. This section explores two approaches to compare the security of wM-Bus communications utilizing Transport Layer Security (TLS) vs the use of noise frames. Table 6.2 offers a comparison between the two protocols.

6.2.1 Memory Usage

1. Traditional TLS 1.2
    - TLS 1.2 relies on complex cryptographic algorithms and a full-featured protocol stack, which can be memory-intensive. This includes the storage of certificates, private keys, and session state information.
    - The memory footprint can be significant, particularly for devices with constrained resources, due to the overhead of cryptographic libraries and the TLS protocol itself.
2. NPF (Pattern XX)
    - The NPF, especially with Pattern XX, is designed to be lightweight and efficient. It typically requires less memory than TLS 1.2 because it uses streamlined cryptographic operations and lacks the extensive protocol overhead of TLS.
    - Pattern XX's use of ChaCha20-Poly1305 and X25519 is optimized for minimal memory usage, making it well-suited for devices with limited resources.

6.2.2 Packet Size

1. Traditional TLS 1.2
    - TLS 1.2 adds a considerable amount of overhead to each packet, including encryption, MAC for integrity, and potentially TLS record headers. This increases

the packet size, which can be a concern in bandwidth-constrained environments.

- The initial handshake also involves multiple round-trips and the exchange of certificate data, which can lead to large packet sizes early in the communication.

2. NPF (Pattern XX)

    - Noise tends to have lower overhead per message compared to TLS, as it's designed for simplicity and efficiency. Pattern XX, in particular, focuses on compact cryptographic primitives, reducing the overall packet size.
    - The handshake process is streamlined, involving fewer messages and less data exchange, leading to smaller initial packet sizes.

6.2.3 Handshake Time

1. Traditional TLS 1.2

    - The TLS 1.2 handshake is a multi-step process that involves several round-trips between the client and server. This can introduce significant latency, especially in networks with high round-trip times.
    - The process of negotiating cryptographic parameters, exchanging certificates, and verifying them can be time-consuming, particularly on devices with limited processing power.

2. NPF (Pattern XX)

    - The Noise handshake is designed to be fast and efficient. Pattern XX, with its choice of cryptographic algorithms, is optimized for quick handshakes with minimal computational overhead.
    - The reduced number of round-trips and the absence of certificate exchanges in the Noise handshake process contribute to shorter completion times.

6.2.4 Summary

Choosing between Traditional TLS 1.2 and the NPF using Pattern XX for securing wM-Bus communications in IoT environments involves balancing security needs with device constraints. While TLS 1.2 offers a well-established and universally supported security solution, its memory and bandwidth requirements, along with the time-consuming handshake

Table 6.2: TLS Vs Noise Framework to Secure Wireless M-Bus

Feature	**TLS 1.2 for Wireless M-Bus**	**Noise Framework (Pattern XX) for Wireless M-Bus**
Security Level	High (cryptographic security with well-established standards)	High (strong cryptographic security with efficient algorithms)
Computational Overhead	High (encryption/decryption, certificate handling)	Moderate (optimized encryption/decryption with less complex handshake)
Network Overhead	Moderate (increased packet size due to encryption overhead and handshake)	Low (streamlined protocol with minimal overhead)
Energy Consumption	Higher (due to more intensive cryptographic operations and longer handshake)	Lower (efficient cryptographic algorithms and faster handshake reduce energy use)
Implementation Complexity	High (requires comprehensive cryptographic libraries and certificate management)	Moderate (simpler protocol, but requires careful implementation of cryptographic primitives)

process, may not be ideal for all IoT devices, especially those with stringent power and resource limitations. On the other hand, the NPF with Pattern XX provides a lightweight and efficient alternative, potentially better suited for constrained environments, with the trade-off being the need for careful implementation and potentially less universal protocol support.

6.3 STRIDE Threat Models

Smart meters are vulnerable to various cyber threats that can cause significant disruptions or data breaches. These threats to the smart metering system can be classified as either active or passive. While passive threats involve merely observing and analyzing the data without any modification, active threats involve the manipulation and alteration of the data. Based on the analysis presented in Chapter 5 along with the specific requirements for building a STRIDE threat model for wM-Bus using the NPF with XX and NX patterns. Therefore, I will outline a theoretical approach based on the given requirements and the security properties of the XX and NX patterns. However, to build a STRIDE threat model for your proposed protocol with the NPF, we need to systematically analyze each component of STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege) in the context of the security properties of the NPF patterns.

6.3.1 STRIDE Threat Models Overview

1. **Spoofing Identity:** This threat involves an attacker impersonating another user or device. In the context of the NPF, the use of secure key exchange mechanisms like X25519 for establishing a shared secret can mitigate this threat. Implementing mutual authentication mechanisms, possibly through a pre-shared key, can help address this threat.

2. **Tampering:** This threat involves an attacker modifying data in transit. The NPF employs encryption algorithms like ChaCha20-Poly1305 AEAD to protect the integrity and confidentiality of the data. Ensuring all data is encrypted and authenticated before transmission can mitigate tampering threats.
3. **Repudiation:** This involves an entity denying the action they performed. To counter this threat, implementing non-repudiation measures such as digital signatures or comprehensive logging mechanisms that record key exchanges and data transactions within the protocol operation can be effective.
4. **Information Disclosure:** This threat concerns unauthorized access to data. The use of strong encryption algorithms like ChaCha20-Poly1305 AEAD ensures that data is encrypted, mitigating the risk of information disclosure. It's crucial to ensure that all sensitive data is encrypted and keys are securely managed.
5. **Denial of Service (DoS):** This threat involves an attacker preventing legitimate users from accessing services. While the NPF primarily focuses on confidentiality and integrity, DoS mitigation might involve implementing rate limiting, authentication before resource allocation, and monitoring abnormal traffic patterns.
6. **Elevation of Privilege:** This threat involves an attacker gaining higher-level permissions than intended. Proper role-based access control and ensuring that the protocol does not inadvertently allow for escalation of privileges through flaws in its design or implementation can mitigate this threat.

Table 6.3: STRIDE Threat Models

Threat	**Desired property**	**Threat Definition**
Spoofing	Authenticity	Pretending to be something or someone other than yourself
Tampering	Integrity	Modifying something on disk, network, memory, or elsewhere
Repudiation	Non-repudiability	Claiming that you didn't do something or were not responsible; can be honest or false
Information disclosure	Confidentiality	Providing information to someone not authorized to access it
Denial of service	Availability	Exhausting resources needed to provide service
Elevation of privilege	Authorization	Allowing someone to do something they are not authorized to do

6.3.2 Define the Components

1. Meter: The device collecting utility usage data.
2. Gateway: The receiver and aggregator of data from one or more meters.
3. Communication Channel: The wireless medium used for data transmission.

4. Pattern XX or NX: A cryptographic pattern from the NPF, tailored for the Meter-Gateway communication.
5. Adversary: An entity attempting to compromise the communication.

6.3.3 Threat Model for NX Pattern

Pattern NX is characterized by:

- The initiator sending an ephemeral public key and encrypted payloads without authenticating itself in the first message.
- The responder authenticating itself to the initiator in the subsequent message as depicted in 6.4

Table 6.4: The handshake NX pattern messages

XX:
→ e
← e, ee, s, es

The integration of the NPF using the NX pattern into wM-Bus communication ensures the confidentiality, integrity, and authenticity of the messages exchanged between the meter and the gateway, under the following conditions:

1. **Theorem: Security of wM-Bus with NPF Pattern XX**
   (a) **Lemma 1 (Confidentiality):** The Noise NX pattern ensures that all communications between the meter and the gateway are encrypted, making it computationally infeasible for unauthorized parties to access the content of the messages without the appropriate private keys.
   (b) **Lemma 2 (Authentication):**The Noise NX pattern provides unilateral authentication of the initiator (usually the meter) to the responder (the gateway), ensuring that the messages originate from a legitimate source.
   (c) **Lemma 3 (Resistance to Replay Attacks:** The Noise NX pattern includes nonce values in its encryption process, making replay attacks ineffective as the same nonce cannot be used more than once without detection.
2. **Games for Security Analysis**

   (a) **Game 1 (Passive Eavesdropping):** An adversary attempts to gain unauthorized access to the communication by passive eavesdropping. Due to the encryption provided by the Noise NX pattern, the adversary cannot derive any meaningful information from the intercepted messages.

   (b) **Game 2 (Impersonation Attacks)** An adversary tries to impersonate a legitimate meter to send fabricated messages to the gateway. However, due to the authentication mechanism in the NX pattern, the gateway can verify the authenticity of the incoming messages, thwarting the impersonation attempt.

   (c) **Game 3 (Replay Attacks):** An adversary captures a legitimate message and attempts to resend it to the gateway. The replay of the message is detected and rejected due to the nonce mechanism in the NX pattern, which ensures that old or replayed messages are identified as invalid.

3. **Conclusion** Employing the NPF's NX pattern in wM-Bus communications significantly enhances security by ensuring that messages are encrypted, authenticated, and protected against replay attacks. This theorem, supported by the specified lemmas and analyzed through the described games, provides a structured framework for understanding the security benefits of integrating the NX pattern into wM-Bus systems, thereby mitigating various potential security threats.

6.3.4 Threat Model for XX Pattern

Pattern XX is a handshake pattern within the NPF that provides mutual authentication and encrypted transport 6.5. This pattern involves:

- Two-way authentication, where both parties prove their identities to each other.
- Key exchange, allowing both parties to derive a shared secret key for encryption and decryption of their communication.

Table 6.5: The handshake XX pattern messages

XX:
→ e
← e, ee, s, se
→ s, es

The security of wM-Bus communication enhanced by the NPF using pattern XX ensures confidentiality, integrity, and authentication of messages between communicating parties, under the assumptions of:

1. **Theorem: Security of wM-Bus with NPF Pattern XX**
    (a) **Lemma 1 (Confidentiality):** Given the NPF's encryption mechanisms, particularly with pattern XX utilizing X25519 for key exchange and ChaCha20-Poly1305 for encryption, it is computationally infeasible for an adversary to decrypt the communication without possession of the private keys.

    (b) **Lemma 2 (Integrity and Authentication):** With the inclusion of a MAC (Message Authentication Code) provided by Poly1305 in the NPF pattern XX, any alterations to the transmitted messages are detectable, ensuring message integrity and authenticating the message source.

    (c) **Lemma 3 (Forward Secrecy):** Each session key is ephemeral and derived from a new set of Diffie-Hellman key exchanges (X25519), ensuring that the compromise of long-term keys does not compromise past session keys.

2. **Games for Security Analysis**
    (a) **Game 1 (Eavesdropping):** An adversary intercepts the communication but cannot derive any meaningful information due to the encryption provided by ChaCha20-Poly1305.

    (b) **Game 2 (Man-in-the-Middle Attack)** An adversary attempts to intercept and alter the messages between parties. The integrity checks and authentication mechanisms prevent unnoticed alterations and impersonation.

    (c) **Game 3 (Key Compromise Impersonation):** An adversary obtains a long-term private key of one party but cannot impersonate them in new sessions due to the ephemeral nature of the session keys and mutual authentication.

3. **Conclusion** The integration of the NPF pattern XX into wM-Bus communication significantly enhances the security by ensuring encrypted, authenticated, and integrity-checked messaging with forward secrecy, assuming the underlying cryptographic primitives remain secure against computational attacks. This theorem and its supporting lemmas and games provide a structured analysis of the security properties introduced by implementing the NPF pattern XX in wM-Bus systems, addressing potential threats and affirming the protocol's resilience against various attack vectors.

6.3.5 Noise Explorer

In the exploration of integrating wM-Bus technology with the NPF, the Noise Explorer [7], a verification tool, was employed to evaluate the security efficacy of two predefined communication patterns, XX and NX, within this framework. The Noise Explorer is instrumental in analyzing cryptographic protocols, focusing on their ability to secure communications through a series of authentication and confidentiality queries. These queries are designed to assess how well a protocol can authenticate parties to each other and maintain the confidentiality of their communications.

The NX pattern demonstrated partial success in the evaluation, successfully passing 2 out of 4 authentication queries and 2 out of 5 confidentiality queries. This indicates that while the NX pattern provides a level of security, it exhibits certain limitations in fully ensuring both authentication and confidentiality across all tested queries. On the other hand, the XX pattern excelled in the assessment, successfully passing all authentication and confidentiality queries posed by the Noise Explorer. This comprehensive success suggests that the XX pattern provides a robust security framework capable of effectively authenticating parties and maintaining the confidentiality of their communications within the scope of the NPF. Sample view of the tool depicted in Figure 6.1

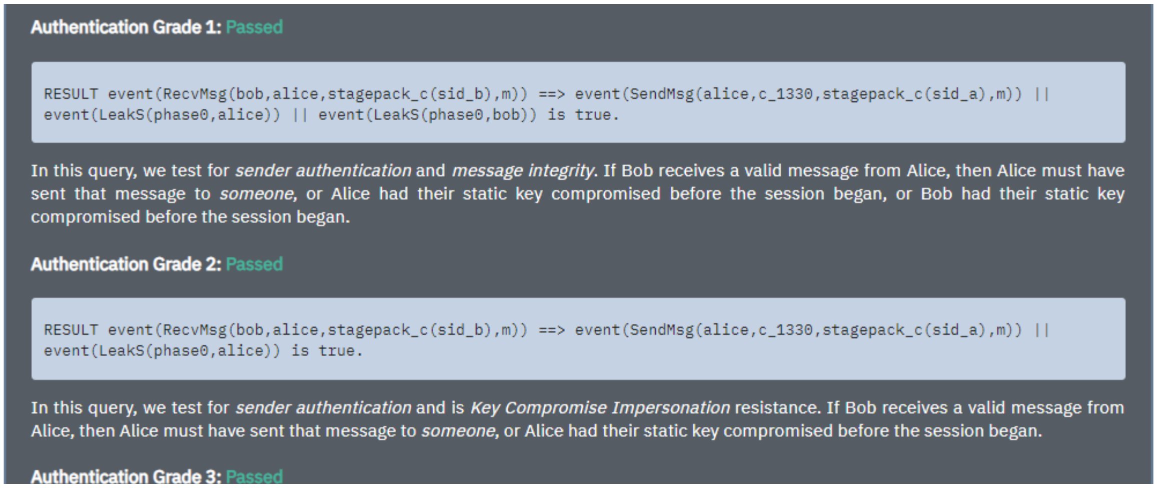
The Noise Explorer tool, in this context, serves as a critical component in assessing the security posture of communication patterns under the NPF. By systematically evaluating the patterns' responses to a series of security queries, the tool provides valuable insights into their strengths and potential vulnerabilities, enabling informed decisions about their deployment in secure communication channels.

6.4 Battery Life Analysis

This section will detail the impact of the chosen security protocols NX, XX and TLS on the battery life of IoT devices. Factors to consider will include the energy consumption associated with cryptographic operations, the impact of increased packet sizes on radio usage, and the overall operational efficiency of the protocol in a low-power environment.

6.4.1 Power Consumption and Battery Life

In evaluating the battery life of IoT devices across three communication protocols; NX, XX, and TLS it's evident that the choice of protocol significantly influences operational longevity. The NX protocol, with its efficient data handling and handshake processes,



Authentication Grade 1: Passed

```
RESULT event(RecvMsg(bob,alice,stagepack_c(sid_b),m)) ==> event(SendMsg(alice,c_1330,stagepack_c(sid_a),m)) ||
event(LeakS(phase0,alice)) || event(LeakS(phase0,bob)) is true.
```

In this query, we test for *sender authentication* and *message integrity*. If Bob receives a valid message from Alice, then Alice must have sent that message to *someone*, or Alice had their static key compromised before the session began, or Bob had their static key compromised before the session began.

Authentication Grade 2: Passed

```
RESULT event(RecvMsg(bob,alice,stagepack_c(sid_b),m)) ==> event(SendMsg(alice,c_1330,stagepack_c(sid_a),m)) ||
event(LeakS(phase0,alice)) is true.
```

In this query, we test for *sender authentication* and is *Key Compromise Impersonation* resistance. If Bob receives a valid message from Alice, then Alice must have sent that message to *someone*, or Alice had their static key compromised before the session began.

Authentication Grade 3: Passed

Figure 6.1: Sample of Noise Explorer Verification Tool for XX pattern

promises the longest battery life of approximately 9.51 years as depicted in Figure 6.2 and Table 6.6. In contrast, the XX protocol, while still efficient, offers a reduced lifespan of about 7.88 years due to slightly higher energy demands. The TLS protocol, known for its robust security features, incurs a substantial energy overhead, limiting the device's operational life to just 3.81 years. These figures highlight the trade-offs between energy efficiency and functionality in IoT communications protocols, with NX emerging as the most energy-efficient option in this analysis.

Table 6.6: Battery life (year)

Protocol	**Battery Life (year)**	**Reduction Percentage at 10 years**	**Reduction Percentage at 12 Years**
wM-Bus + NX	9.51	5%	20.83%
wM-Bus + XX	7.88	21.2%	34.33%
wM-Bus + TLS	3.81	61%	67.67%

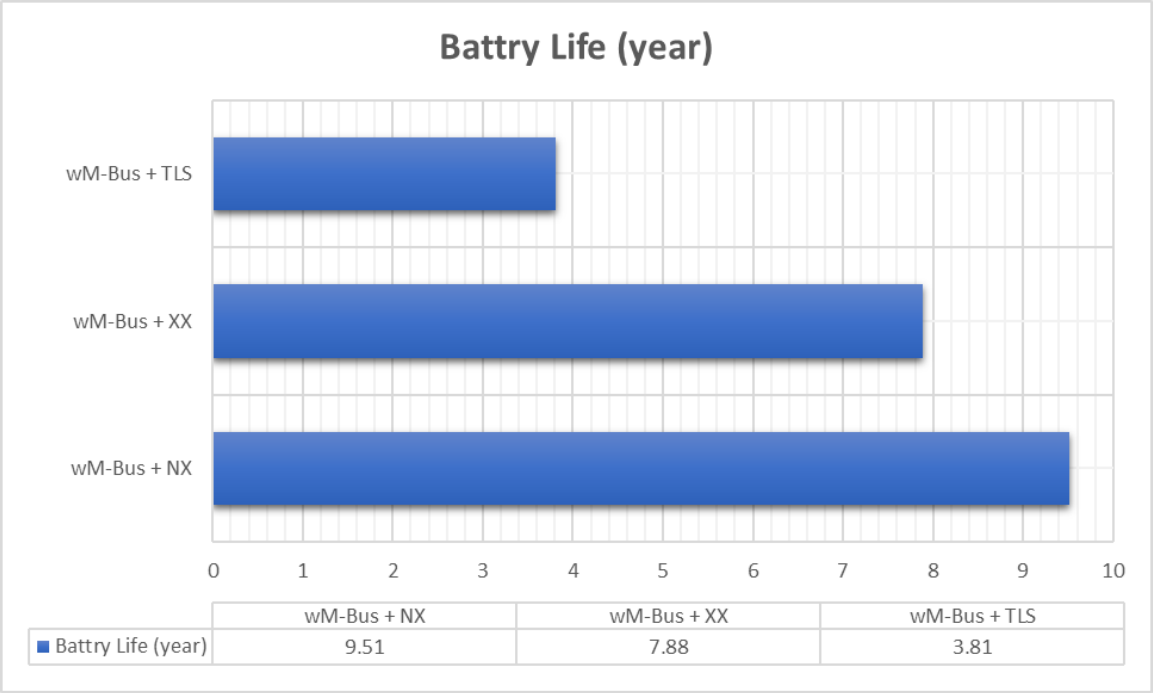


Figure 6.2: Battery life for the different Protocol

6.4.2 Battery Life Calculation for NX Pattern

Calculating the power consumption for a specific framework or software package like NPF pattern NX and XX using a meter device with the model RC1701HP-MBUS4 requires more specific information. Power consumption depends on various factors, including the hardware its running it on, the usage patterns, and the workload. However, relying on the datasheet of the RC107HP-MBUS4 chip help in estimated the power consumption and the battery life using theses two patterns. Furthermore, insights from our practical implementation enrich these estimations. first we generated the following assumptions:

- Data transmission occurs every 12 hour.
- Data rate is 19.2 kbps.
- Transmission power is increased to 30 dBm.
- Duty cycle is 0.1% (0.001 as a decimal).
- Battery capacity: 2000 mAh.
- Current consumption, RX/IDLE: 31.7 mA

- Current consumption, TX (+27/30 dBm): 703 mA
- Current consumption, SLEEP: Max 2.0 uA (microamperes)
- Handshake Elapsed Time: 18.17276549
- Packet Size: 1278 Bytes
- Memory Usage: 9447

1. **Time Between Transmissions**

   With data transmissions occurring twice a day, the time between transmissions is 12 hours. This is because there are 24 hours in a day, so with two transmissions, each occurs every 12 hours.

2. **Handshake Energy Consumption**

   The handshake process is assumed to consume energy at the RX/IDLE rate, given that it involves communication but not full data transmission. The energy consumed during the handshake process is calculated as:

$$\begin{aligned}\text{Handshake Energy Consumption(mAh)} &= \text{Current Consumption} \times \frac{\text{Handshake Time(s)}}{3600} \\ &= 31.7 \times \frac{18.17276549}{3600}\ \text{mAh} \\ &\approx 0.159 mAh \end{aligned} \tag{6.1}$$

3. **Total Active Time**

   The NX packet transmission time was calculated based on a packet size of 1278 bytes and a data rate of 19.2 kbps. The total active time includes both the handshake time and the packet transmission time:

$$\begin{aligned}\text{NX Packet Transmission Time(s)} &= \frac{1278 \times 8\,\text{bits}}{19.2 \times 1000\,\text{bps}} \\ &\approx 0.67083333\,\text{s}\end{aligned} \tag{6.2}$$

$$\text{Total Active Time(s)} = \text{Handshake Time(s)} + \text{NX Packet Transmission Time (s)}$$

$$\text{Total Active Time(s) } = 18.17276549 s + \approx 0.67083333 \tag{6.3}$$

4. **NX Packet Transmission Energy Consumption**

   The energy consumption during the total active time includes energy for the handshake and for transmitting the packet:

$$E_{\text{TX}} = 703\,\text{mA} \times \frac{\text{NX packet transmission time (s)}}{3600} \tag{6.4}$$

Substituting into E_{TX}:

$$E_{\text{TX}} \approx 703\,\text{mA} \times \frac{0.67083333\,\text{s}}{3600} \approx 0.104\,\text{mAh} \tag{6.5}$$

5. **Total Active Energy Consumption**

$$E_{\text{active}} = 0.159\,\text{mAh} + 0.104\,\text{mAh} \approx 0.264\,\text{mAh} \tag{6.6}$$

6. **Adjusted Sleep Mode Time**
   With 12-hour intervals between transmissions, the adjusted sleep mode time is:

$$\begin{aligned} &\text{Adjusted sleep mode time (s)} = 12 \times 3600 - \text{Total active time (s)} \\ &\text{Adjusted sleep mode time (s)} \approx 12 \times 3600 - (18.17276549 + 0.67083333) \\ &\approx 43161.156\,\text{s} \end{aligned} \tag{6.7}$$

7. **Energy Consumption in Sleep Mode**

   Using the sleep mode current consumption rate of 2.0 µA (which is 2.0×10^{-3}mA):

$$\text{Sleep mode energy consumption (mAh)} = 2.0 \times 10^{-3}\,\text{mA} \times \frac{43161.156\,\text{s}}{3600} \approx 0.024\,\text{mAh} \tag{6.8}$$

8. **Total Energy Consumption per NX Cycle**
   The total energy consumption per NX cycle, including the handshake, is the sum of active and sleep mode energy consumption:

$$\text{Total energy consumption per cycle (mAh)} = 0.264\,\text{mAh} + 0.024\,\text{mAh} \quad \approx 0.288\,\text{mAh} \tag{6.9}$$

9. **Number of Cycles and Operational Lifetime**
   The number of cycles the battery can support is calculated by dividing the battery

capacity by the total energy consumption per cycle:

$$\text{Number of cycles} = \frac{2000\,\text{mAh}}{0.288\,\text{mAh/cycle}} \approx 6944.44\,\text{cycles} \tag{6.10}$$

The total operational lifetime in days and then in years is:

$$\text{Total operational lifetime (days)} = \frac{6944.44\,\text{cycles} \times 12\,\text{hours/cycle}}{24} \approx 3472.22\,\text{days} \tag{6.11}$$

$$\text{Total operational lifetime (years)} = \frac{3472.22\,\text{days}}{365} \approx 9.51\,\text{years} \tag{6.12}$$

Based on the provided assumptions, and the detailed breakdown of how we arrived at the operational lifetime of approximately **9.51 years** with twice-daily transmissions, including the handshake time.

6.4.3 Battery Life Calculation for XX Pattern

Battery life calculated based on the following assumptions:

- Data transmission occurs every 12 hour.
- Data packet size for the handshake is 1500 bytes.
- Data rate is 19.2 kbps.
- Transmission power is increased to 30 dBm.
- Duty cycle is 0.1% (0.001 as a decimal).
- Battery capacity: 2000 mAh.
- Current consumption, RX/IDLE: 31.7 mA
- Current consumption, TX (+27/30 dBm): 703 mA
- Current consumption, SLEEP: Max 2.0 uA (microamperes)
- Handshake Elapsed Time: 22.921872
- Memory Usage: 14972

For the XX pattern with the provided assumptions, and following the same calculations as NX, the calculations yield the following results:

- The total energy consumption per transmission cycle is approximately 0.348 mAh.
- The device can support around 5749 transmission cycles with a 2000 mAh battery.
- Consequently, the total operational lifetime is about **7.88 years**, considering data transmission occurs every 12 hours.

These results demonstrate the impact of the handshake time and data packet size on the device's energy consumption and overall operational lifetime

6.4.4 Battery Life Calculation for TLS

- Data transmission occurs every 12 hour.
- Data packet size for the handshake is 6449 bytes.
- Data rate is 19.2 kbps.
- Transmission power is increased to 30 dBm.
- Duty cycle is 0.1% (0.001 as a decimal).
- Battery capacity: 2000 mAh.
- Current consumption, RX/IDLE: 31.7 mA
- Current consumption, TX (+27/30 dBm): 703 mA
- Current consumption, SLEEP: Max 2.0 uA (microamperes)
- Handshake Elapsed Time: 19.33999
- Memory Usage: 897,2897

For the TLS pattern with the specified assumptions, the calculations yield the following results:

- The total energy consumption per transmission cycle is approximately 0.719 mAh.
- The device can support around 2782 transmission cycles with a 2000 mAh battery.
- Consequently, the total operational lifetime is about **3.81 years**, considering data transmission occurs every 12 hours.

Chapter 7

7 Conclusion and Future Work

In the era of digital transformation, Internet of Things (IoT) devices have become integral to our daily lives, influencing various aspects such as home automation, healthcare monitoring, industrial control systems, and smart cities. The pervasive nature of these devices underscores the critical importance of security in safeguarding our personal data, privacy, and the seamless functionality of the systems we rely on. However, despite the undeniable need for robust security measures, a notable gap persists in the IoT landscape. Certain applications, prioritizing cost reduction, ease of deployment, and energy efficiency, often overlook the implementation of adequate security protocols. This oversight is particularly pronounced in battery-powered IoT devices, such as smart meters for water and gas, where the additional energy consumption attributed to security measures is a significant concern.

Smart meters, which are pivotal in modernizing utility management and enhancing energy efficiency, epitomize this challenge. These devices, especially those monitoring water and gas consumption, are typically battery-operated and are expected to function for extended periods without maintenance. The introduction of any security mechanism that could potentially drain the battery or complicate the deployment process is often met with resistance, highlighting a critical trade-off between security and operational efficiency.

This study introduces the novel application of the NPF to secure wM-Bus communications, which are pivotal for the smart metering infrastructure. Recognizing the vital role of security in the reliable and safe operation of IoT devices, this research aims to bridge the gap by offering a security model does not compromise device longevity or deployment simplicity.

7.1 Summary of Contributions

The core contributions of this research are outlined as follows:

1. Comprehensive Study of Wireless Communication Technologies: A comprehensive review of 12 wireless communication protocols was undertaken to identify the most suitable technology for AMI, considering both operational and security requirements. Mostly, focusing on key architectural elements like range, data rate, frequency, protocol structure, and security.

   - Extensive Exploration: Organized technologies based on their communication range into three groups:

Short Range	**Medium Range**	**Long Range**
WPAN & WHAN	**WLAN**	**WWAN**
Bluetooth & BLE	Wi-Fi (802.11a/b/g/n)	LoRa
ZigBee & ZigBee PRO	Wi-Fi HaLow (802.11ah)	Sigfox
Z-Wave & Z-Wave Plus	Wi-SUN	LTE-M
Dash7	wM-Bus	NB-IoT
RPMA		

   - Choosing wM-Bus and Addressing its Security Challenges: wM-Bus was ultimately selected due to its relevance and widespread adoption in the AMI landscape. The research then focused on addressing its security vulnerabilities. Through detailed analysis and strategic intervention, the study enhanced the security posture of AMI systems, leveraging wM-Bus's strengths while mitigating its security risks.

2. Implementation of the NPF to Secure wM-Bus: This research segment highlights the innovative integration of the NPF as a security measure for wM-Bus systems. This initiative represents a significant advancement in merging of the Noise Protocol with the operational framework of wM-Bus, thereby establishing enhanced security standards within the IoT ecosystem. The comprehensive implementation of the NPF for securing wM-Bus communications was systematically carried out in five phases:
   - Preliminary Implementation of Noise Protocol: Initiated with the implementation of the NPF, incorporating six predefined patterns to establish a foundational benchmark for subsequent phases.
   - Preliminary wM-Bus Implementation: Involved in setting up a simulated environment representing meters and gateways to implement wM-Bus communication, thereby creating a benchmark for wM-Bus performance and functionality.

  - TLS Implementation for wM-Bus: Focused on securing wM-Bus communications using the Transport Layer Security (TLS) protocol. This phase aimed to establish a benchmark for evaluating the security and performance implications of using TLS in the AMI context.
  - Noise Protocol Implementation for wM-Bus: This phase involved the actual integration of the NPF to secure wM-Bus communications, with a focus on assessing the enhancements in security and system performance, thereby establishing a new benchmark.
  - Optimization of Noise Framework: Aimed at refining and optimizing the Noise Protocol implementation for wM-Bus, this phase focused on fine-tuning the framework to maximize security efficiency and system performance within the AMI ecosystem.

3. Comprehensive Evaluation: The evaluation summary of the research involving the NPF and its comparison with TLS for securing wM-Bus communications is as follows:
    - Evaluation of Noise Framework Patterns: The research evaluated 12 optimized NPF patterns, focusing on memory usage, packet size, and handshake elapsed time. The NX and XX patterns emerged as the most efficient, with NX being a two-message pattern and XX a three-message pattern, demonstrating low memory usage, small packet sizes, and shorter handshake times.
    - Comparative Analysis with TLS: A detailed comparison highlighted the operational advantages and enhanced security features of the NPF over TLS. The NX and XX patterns were specifically analyzed, showing significant improvements in operational lifetime and resource utilization compared to TLS.
    - Security Evaluation of NX and XX Patterns: The security of both NX and XX patterns was rigorously evaluated using the STRIDE threat model, and the Noise Explorer tool was used to affirm their security. The NX pattern showed partial success in Noise Explorer evaluation, while the XX pattern successfully passed all authentication and confidentiality queries, indicating a robust security framework.
    - Battery Life Evaluation: The battery life for IoT devices using NX, XX, and TLS protocols was analyzed, demonstrating the lightweight nature of the Noise Framework. NX protocol offered the longest battery life of approximately 9.51

years, XX protocol provided around 7.88 years, and TLS significantly reduced battery life to 3.81 years. This showcases the efficiency of NX and XX patterns in terms of energy consumption compared to TLS.

These findings underline the effectiveness of the NPF, particularly the NX and XX patterns, in providing a secure and efficient solution for wM-Bus communications, outperforming the TLS in terms of resource utilization and operational longevity.

Therefore, this research has achieved the objectives mentioned in Chapter 1 :

- Development of a Secure and Lightweight Protocol: The thesis successfully developed a secure, lightweight wM-Bus protocol leveraging the NPF, which significantly reduces computational and power requirements without compromising the security and integrity of communications in IoT environments.
- Enhancement of Security and Privacy: The newly developed protocol addresses the key security challenges of confidentiality, integrity, and authentication, which are crucial for IoT devices like smart meters. This aligns with the goals to enhance security measures for IoT communications.
- Improvement in Energy Efficiency: One of the central aims was to improve energy efficiency in the operation of IoT devices. The protocol developed demonstrates superior energy efficiency by extending the operational lifetime of devices compared to existing solutions like TLS, thus meeting the objective of reducing energy consumption.
- Robustness Against Security Threats: The protocol's design includes mechanisms to protect against various security threats, effectively balancing protection against sophisticated threats with the need for efficiency and optimal performance. This is in line with the objective to develop a protocol that could safeguard IoT devices from an array of cybersecurity risks.
- Practical and Theoretical Contributions: The thesis provides both practical implementations and theoretical frameworks, offering a comprehensive approach to securing IoT communications. This fulfills the objective of contributing practically applicable and theoretically robust solutions to the field.

7.2 Research Limitation

There are several key limitations and considerations of research that focused on developing a secure communication protocol using the NPF within a wM-Bus communication setting. Here's a summary of the key points:

1. Device Limitation: The study was based on the use of only two specific devices to simulate the communication between the meter and the gateway. This limited simulation may not fully represent real-world complexities and could affect the generalizability of the findings.
2. Design Constraints: The protocol was designed to balance security with practicality and efficiency within the confines of the Wireless M-Bus communication. This required choosing compatible protocol patterns with the existing hardware and software, potentially limiting flexibility and adaptability to systems with different specifications.
3. Communication Standard Constraints: The protocol was aligned to fit within the specific constraints of the Wireless M-Bus network packet frame, influencing decisions like the length of messages and excluding certain cipher suites and security properties that would have required more extensive data exchanges.

7.3 Future Research Directions

The research presents a promising foundation for enhancing IoT security through the NPF, suggesting several avenues for future exploration. Firstly, the broader application of the NPF across various IoT ecosystems represents an exciting frontier. The study not only advocates for its widespread adoption but also proposes further optimization of Noise patterns to improve security and efficiency in diverse IoT protocols and environments. This initiative could significantly elevate the standard of IoT security, making it more adaptable and robust against emerging threats.

Secondly, transitioning from theoretical models to practical implementation holds substantial potential. Collaborating with a metering company to integrate the optimized NPF directly into meter hardware could provide invaluable insights into real-world applicability and performance. This hands-on approach would not only validate the theoretical findings but also refine the framework based on practical challenges and operational feedback, ensuring that the security solutions are both effective and seamlessly compatible into existing infrastructures.

In addition, our solution is not quantum-ready. While current cryptographic mechanisms provide robust security for traditional computing environments, they may not withstand the advanced computational capabilities of quantum computers. Our proposed solution, which relies on these traditional cryptographic techniques, may become vulnerable once quantum computing becomes more prevalent. Future research should explore integrating quantum-resistant algorithms to ensure long-term security in the evolving technological landscape

Lastly, addressing the limitations identified in the study, particularly concerning message size constraints, offers a critical path for future work. By testing the NPF alongside the latest TLS 1.3, and experimenting with longer message formats, the research can push the boundaries of current implementations. This effort would not only benchmark Noise Framework's capabilities against the most advanced TLS version but also explore solutions to overcome limitations related to message sizes, thereby enhancing the framework's versatility and applicability in more complex scenarios.

Bibliography

[1] OMS-Group. “M-bus official website.” (2024), [Online]. Available: `https://m-bus.com/` (visited on 04/15/2024).

[2] T. Perrin, “The noise protocol framework,” *PowerPoint Presentation*, 2018. [Online]. Available: `https://noiseprotocol.org/noise.html`.

[3] W. Anani, A. Ouda, and A. Hamou, “A survey of wireless communications for iot echo-systems,” in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, IEEE, 2019, pp. 1–6.

[4] W. Anani and A. Ouda, “Demystifying wireless technologies for best uses in iot echo-systems,” in *2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH)*, IEEE, 2022, pp. 238–245.

[5] W. Anani and A. Ouda, “Wireless meter bus: Secure remote metering within the iot smart grid,” in *2022 International Symposium on Networks, Computers and Communications (ISNCC)*, IEEE, 2022, pp. 1–6.

[6] Radiocrafts. “An024:wirelessm-bus inindustrialsensornetworks.” (), [Online]. Available: `https://radiocrafts.com/uploads/AN024_Using_Wireless_M-Bus_in_Industrial_Sensor_Networks.pdf`. accessed: 01.01.2017.

[7] N. Kobeissi, G. Nicolas, and K. Bhargavan, *Noise explorer: Fully automated modeling and verification for arbitrary noise protocols*, Cryptology ePrint Archive, Paper 2018/766, `https://eprint.iacr.org/2018/766`, 2018. [Online]. Available: `https://eprint.iacr.org/2018/766`.

[8] L. Kohnfelder and P. Garg, “The threats to our products (1999),” *URL: https://adam. shostack. org/microsoft/The-Threats-To-Our-Products. docx*, 2021.

[9] R. S. Adhikari, N. Aste, and M. Manfren, “Multi-commodity network flow models for dynamic energy management–smart grid applications,” *Energy Procedia*, vol. 14, pp. 1374–1379, 2012.

[10] Ç. Iris and J. S. L. Lam, “Optimal energy management and operations planning in seaports with smart grid while harnessing renewable energy under uncertainty,” *Omega*, vol. 103, p. 102 445, 2021.

[11] E. al. N.S. Gowri Ganesh, “Bql-drs: A novel balanced q-learning based demand response system for iot based smart grids,” *International Journal on Recent and Innovation Trends in Computing and Communication*, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:265864176`.

[12] B. Kazemi, A. Kavousi-fard, M. Dabbaghjamanesh, and M. Karimi, “Iot-enabled operation of multi energy hubs considering electric vehicles and demand response,”

IEEE Transactions on Intelligent Transportation Systems, vol. 24, pp. 2668–2676, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:246349963`.

[13] K. Dewri, M. Hasan, and M. R. Uddin, “Iot-based energy optimization and demand response system for renewable energy integration,” *2023 10th IEEE International Conference on Power Systems (ICPS)*, pp. 1–5, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:267659786`.

[14] M. Jamjoum, R. Negry, E. Abdelsalam, F. Almomani, T. Salameh, and A. A. Makky, “Customers dependent demand response in energy hubs with iot perforation,” *2023 Advances in Science and Engineering Technology International Conferences (ASET)*, pp. 1–5, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:260003215`.

[15] M. Saravanan, A. Das, and V. Iyer, “Smart water grid management using lpwan iot technology,” *2017 Global Internet of Things Summit (GIoTS)*, pp. 1–6, 2017. [Online]. Available: `https://api.semanticscholar.org/CorpusID:12251080`.

[16] Q. V. Khanh, N. V. Hoai, L. D. Manh, A. N. Le, and G. Jeon, “Wireless communication technologies for iot in 5g: Vision, applications, and challenges,” *Wireless Communications and Mobile Computing*, 2022. [Online]. Available: `https://api.semanticscholar.org/CorpusID:246655298`.

[17] V. P. Delsin, A. N. L. Junior, and R. Barbosa, “Smart grid system prototype model with lpwan network for communication,” *Journal of Engineering Research*, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:264109537`.

[18] Y. Li, B. Adili, Z. Zhang, Z. Ding, L. Li, and C. Chen, “Research on adaption analysis between lpwan and smart grid,” *2023 IEEE 11th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 11, pp. 1967–1971, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:267387759`.

[19] R. Mozny, P. Maek, D. Moltchanov, *et al.*, “Characterizing optimal lpwan access delay in massive multi-rat smart grid deployments,” *Internet of Things*, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:266093707`.

[20] M. Abbasi, S. Khorasanian, and M. H. Yaghmaee, “Low-power wide area network (lpwan) for smart grid: An in-depth study on lorawan,” *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, pp. 022–029, 2019. [On-

line]. Available: `https://api.semanticscholar.org/CorpusID:189824860`.

[21] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.

[22] J. R. A. Jambi, W. K. Wong, F. H. Juwono, and F. Motalebi, "Smart energy meter implementation: Security challenges and opportunities," *2023 International Conference on Digital Applications, Transformation & Economy (ICDATE)*, pp. 1–7, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:261900747`.

[23] C. Portalés, S. Casas, and K. Kreuzer, "Challenges and trends in home automation: Addressing the interoperability problem with the open-source platform openhab," *Harnessing the Internet of Everything (IoE) for Accelerated Innovation Opportunities*, pp. 148–174, 2019.

[24] L. Polcák and P. Matousek, "Metering homes: Do energy efficiency and privacy need to be in conflict?" *SECRYPT*, pp. 47–58, 2022.

[25] N. Kobeissi, G. Nicolas, and K. Bhargavan, "Noise explorer: Fully automated modeling and verification for arbitrary noise protocols," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2019, pp. 356–370.

[26] A. Suter-Dörig, "Formalizing and verifying the security protocols from the noise framework," *Bachelor Thesis*, p. 184, 2018.

[27] A. L. Y. Nir, "Chacha20 and poly1305 for ietf protocols.," Tech. Rep., 2015. [Online]. Available: `http://www.ietf.org/rfc/rfc7539.txt`.

[28] M. J. Saarinen and J.-P. Aumasson, "The blake2 cryptographic hash and message authentication code (mac)," Tech. Rep., 2015. [Online]. Available: `http://www.ietf.org/rfc/rfc7693.txt`.

[29] M. Orlando, A. Estebsari, E. Pons, *et al.*, "A smart meter infrastructure for smart grid iot applications," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12 529–12 541, 2021.

[30] R. P. Díaz Redondo, A. Fernández-Vilas, and G. Fernández dos Reis, "Security aspects in smart meters: Analysis and prevention," *Sensors*, vol. 20, no. 14, p. 3977, 2020.

[31] J. Horalek and V. Sobeslav, "Security baseline for substation automation systems," *Sensors*, vol. 23, no. 16, p. 7125, 2023.

[32] W. Wang and Z. Lu, "Cyber security in the smart grid: Survey and challenges," *Computer networks*, vol. 57, no. 5, pp. 1344–1371, 2013.

[33] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Liatifis, T. Apostolakos, and S. Oikonomou, "An overview of the firewall systems in the smart grid paradigm," in *2018 Global information infrastructure and networking symposium (GIIS)*, IEEE, 2018, pp. 1–4.

[34] D. Kohout, T. Lieskovan, and P. Mlynek, "Smart metering cybersecurity—requirements, methodology, and testing," *Sensors*, vol. 23, no. 8, p. 4043, 2023.

[35] K. Zeman, P. Masek, J. Krejci, *et al.*, "Wireless m-bus in industrial iot: Technology overview and prototype implementation," in *European Wireless 2017; 23th European Wireless Conference*, VDE, 2017, pp. 1–6.

[36] P. Maek, M. Stusek, K. Zeman, R. Mozny, A. Ometov, and J. Hosek, "A perspective on wireless m-bus for smart electricity grids," *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, pp. 730–735, 2019. [Online]. Available: `https://api.semanticscholar.org/CorpusID:198930342`.

[37] F. Derbel, "Smart metering based on automated meter reading," in *2008 5th International Multi-Conference on Systems, Signals and Devices*, IEEE, 2008, pp. 1–2.

[38] K. Akabane, N. Mochizuki, S. Teruhi, *et al.*, "High-capacity wireless access networks using 920mhz band for wide-area iot/m2m services," *IEICE Transactions on Communications*, vol. 99, no. 9, pp. 1920–1929, 2016.

[39] M. Jalasri and L. Lakshmanan, "Managing data security in fog computing in iot devices using noise framework encryption with power probabilistic clustering algorithm," *Cluster Computing*, vol. 26, no. 1, pp. 823–836, 2023.

[40] G. T. Davies, S. Faller, K. Gellert, *et al.*, "Security analysis of the whatsapp end-to-end encrypted backup protocol," in *Annual International Cryptology Conference*, Springer, 2023, pp. 330–361.

[41] B. Dowling, P. Rösler, and J. Schwenk, "Flexible authenticated and confidential channel establishment (facce): Analyzing the noise protocol framework," in *Public-Key Cryptography–PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4–7, 2020, Proceedings, Part I 23*, Springer, 2020, pp. 341–373.

[42] M. R. Palattella, M. Dohler, A. Grieco, *et al.*, "Internet of things in the 5g era: Enablers, architecture, and business models," *IEEE J.Sel. A. Commun.*, vol. 34, no. 3, pp. 510–527, Mar. 2016, ISSN: 0733-8716. DOI: `10.1109/JSAC.2016.2525418`. [Online]. Available: `https://doi.org/10.1109/JSAC.2016.2525418`.

[43] M. S. Mahmoud and A. A. Mohamad, "A study of efficient power consumption wireless communication techniques/modules for internet of things (iot) applications," 2016.

[44] A. Burg, A. Chattopadhyay, and K.-Y. Lam, “Wireless communication and security issues for cyber–physical systems and the internet-of-things,” *Proceedings of the IEEE*, vol. 106, no. 1, pp. 38–60, 2017.

[45] M. Wolf and D. Serpanos, “Safety and security in cyber-physical systems and internet-of-things systems,” *Proceedings of the IEEE*, vol. 106, no. 1, pp. 9–20, 2017.

[46] A. Satan and Z. Toth, “Development of bluetooth based indoor positioning application,” in *2018 IEEE international conference on future IoT technologies (Future IoT)*, IEEE, 2018, pp. 1–6.

[47] C. Prapasawad, K. Pornprasitpol, and W. Pora, “Development of an automatic meter reading system based on zigbee pro smart energy profile ieee 802.15. 4 standard,” in *2012 IEEE International Conference on Electron Devices and Solid State Circuit (EDSSC)*, IEEE, 2012, pp. 1–3.

[48] P. Radmand, M. Domingo, J. Singh, *et al.*, “Zigbee/zigbee pro security assessment based on compromised cryptographic keys,” in *2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, IEEE, 2010, pp. 465–470.

[49] C.-C. Wei, Y.-M. Chen, C.-C. Chang, and C.-H. Yu, “The implementation of smart electronic locking system based on z-wave and internet,” in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, 2015.

[50] E. Nugroho, A. Sahroni, *et al.*, “Zigbee and wifi network interface on wireless sensor networks,” in *2014 Makassar International Conference on Electrical Engineering and Informatics (MICEEI)*, IEEE, 2014, pp. 54–58.

[51] L. Qiao, Z. Zheng, W. Cui, and L. Wang, “A survey on wi-fi halow technology for internet of things,” in *2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2)*, IEEE, 2018, pp. 1–5.

[52] N. S. Knyazev, V. A. Chechetkin, and D. A. Letavin, “Comparative analysis of standards for low-power wide-area network,” in *2017 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SINKHROINFO)*, IEEE, 2017, pp. 1–4.

[53] M. Meera and S. N. Rao, “A survey of the state of the art of 802.11 ah,” in *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, IEEE, 2017, pp. 1–4.

[54] K. Mochizuki, K. Obata, K. Mizutani, and H. Harada, “Development and field experiment of wide area wi-sun system based on ieee 802.15. 4g,” in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, IEEE, 2016, pp. 76–81.

[55] C. Koulamas, S. Giannoulis, and A. Fournaris, "Iot components for secure smart building environments," *Components and services for IoT platforms: Paving the way for IoT standards*, pp. 335–353, 2017.

[56] M. Weyn, G. Ergeerts, L. Wante, C. Vercauteren, and P. Hellinckx, "Survey of the dash7 alliance protocol for 433 mhz wireless sensor communication," *International Journal of Distributed Sensor Networks*, vol. 9, no. 12, p. 870 430, 2013.

[57] A. Sikora, P. Villalonga, and K. Landwehr, "Extensions to wireless m-bus protocol for smart metering and smart grid application," in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, 2012, pp. 399–404.

[58] P. Masek, K. Zeman, Z. Kuder, *et al.*, "Wireless m-bus: An attractive m2m technology for 5g-grade home automation," in *Internet of Things. IoT Infrastructures: Second International Summit, IoT 360° 2015, Rome, Italy, October 27–29, 2015, Revised Selected Papers, Part I*, Springer, 2016, pp. 144–156.

[59] S. Spinsante, S. Squartini, L. Gabrielli, M. Pizzichini, E. Gambi, and F. Piazza, "Wireless m-bus sensor networks for smart water grids: Analysis and results," *International Journal of Distributed Sensor Networks*, vol. 10, no. 6, p. 579 271, 2014.

[60] C. G. Skjæveland, "Infrastructure for collecting and analysing near real-time data from several water meters using wireless m-bus," 2021.

[61] C. Brunschwiler, "Energy fraud and orchestrated blackouts-issues with wireless metering protocols (wm-bus)," *Black Hat USA 2013*, 2013.

[62] OMS. "Open metering system specification, volume 2, primary communication." (), [Online]. Available: `https://oms-group.org/fileadmin/files/download4all/omsSpezifikationen/generation3/spezifikation/vol2/OMS-Spec_Vol2_Primary_v301.pdf`. (accessed: 01.01.2011).

[63] A. Sikora, "Implementation of standardized secure smart meter communication," in *Intelec 2013; 35th International Telecommunications Energy Conference, SMART POWER AND EFFICIENCY*, VDE, 2013, pp. 1–5.

[64] Radiocrafts. "Wireless m-bus products." (2023), [Online]. Available: `https://radiocrafts.com/products/wirelessmbus/`.

Curriculum Vitae

Name:	Wafa Anani
Post-Secondary Education and Degrees:	Ajman University, Ajman, UAE 1988 - 1995 B.Sc University of Western Ontario, London, ON, Canada 2016 - 2016 M.Eng 2017 - 2018 M.E.Sc
Honours and Awards:	Queen Elizabeth II Graduate Scholarships in Science and Technology (QEII-GSST) 2019-2020
Related Work Experience:	Teaching Assistant The University of Western Ontario 2016 - 2023

Publications:

1. Anani W., Ouda A. (2024, March 15). A Secure Lightweight Wireless M-Bus Protocol for IoT: Leveraging the Noise Protocol Framework. (Accepted in IEEE Canadian Journal for Electrical and Computer Engineering.)
2. Anani W., Ouda A. (2022, July). Wireless Meter Bus: Secure Remote Metering within the IoT Smart Grid. In 2022 International Symposium on Networks, Computers and Communications (ISNCC) (pp. 1-6). IEEE.
3. Anani W., Ouda A. (2022, May). Demystifying Wireless Technologies for Best Uses in IoT Echo-Systems. In 2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH) (pp. 238-245). IEEE.
4. Anani W., Ouda, A., & Hamou, A. (2019, May). A survey of wireless communications for IoT echo-systems. In 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE) (pp. 1-6). IEEE.
5. Anani W., Samarabandu, J. (2018, May). Comparison of recurrent neural network algorithms for intrusion detection based on predicting packet sequences. In 2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE) (pp. 1-4). IEEE.
6. Anani W., Ouda A. (2017, April). The importance of human dynamics in the future user authentication. In 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE) (pp. 1-5). IEEE.