People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Abbes Laghrour Khenchela
Faculty of Sciences and Technology
Departement of Mathematics and Computer Science

Order Number
Series:

Thesis for the degree of
Doctorate (LMD)
Specialty: Computer science
Research laboratory: Ingénerie des COnnaissances et Sécurité Informatique (ICOSI)

Security of communications within the Internet of Things context

Presented by

Bettayeb Sami

Jury Members:

Hioual Ouided	MCA - University of Khenchela	President
Hemam Sofiane Mounine	Prof. - University of Khenchela	Director
Messai Mohamed-Lamine	MCA - University of Lyon 2, France	Co-Director
Rahab Hichem	MCA - University of Khenchela	Examiner
Farah Zoubeyr	MCA - University of Bejaia	Examiner
Mechta Djamila	MCA - University of Setif - 1	Examiner

Abstract

The Internet of Things (IoT) has emerged as a transformative force in recent years, reshaping the digital landscape. IoT encompasses a diverse array of interconnected entities, from tiny sensors to large-scale devices, revolutionizing various aspects of daily life. However, this unprecedented growth also brings forth pressing security and privacy concerns that can potentially overshadow the benefits of IoT. Securing these systems poses multifaceted challenges, particularly for key management in resource-constrained and expansive environments. This doctoral thesis endeavors to address the security and privacy challenges inherent to IoT at various layers. We proposed two robust and efficient security schemes tailored to thwart threats originating from cyber realms. The first contribution introduces a new key management scheme that leverages pre-distributed vectors to overcome existing limitations, thereby enhancing security for key establishment, new node addition, refresh, and revocation processes. This scheme further divides the network area into subareas, making our solution not only lightweight, flexible, and scalable but also resilient to multiple attacks. It achieves a substantial reduction in storage requirements, saving more than 81.84%. In terms of communications, during the group-wise key establishment, it achieves 100% efficiency. Furthermore, computation overheads are significantly reduced by reducing the number of multiplication operations by 50%, and energy consumption is optimized, with savings of up to 99.99% during the group-wise key establishment phase. Moreover, the results show that our scheme is more resilient against node capture attacks by up to 96.43% during the initialization phase and by more than 9.89% after, making it suitable for use in resource-limited IoT networks. The second contribution, specifically focusing on key revocation, introduces a groundbreaking solution utilizing blockchain and smart contracts to address the efficiency and security challenges associated with traditional key revocation methods. Our solution reduces communication overhead by 93.55%, 91.87%, and 99.75% compared to other solutions during compromising, leaving, and draining cases, respectively. While BAN (Burrows Abadi Needham) logic plays a pivotal role in the first contribution. This validates the efficiency and appropriateness of our solutions for IoT networks.

***Keywords*:** Internet of Things (IoT), Wireless Sensor Networks (WSN), Security, Key Management, Limited-resource devices, Blockchain, Smart Contracts.

مُلخَّص

انتشرت إنترنت الأشياء (IoT) كقوة تحولية في السَّنوات الأخيرة، مع إعادة تشكيل النَّظرة الرَّقمية. تَضُم IoT مجموعة متنوعة من الكيانات المتصلة، بدءًا من الأجهزة الصَّغيرة إلى الأجهزة الضَّخمة، مُحدِثَة ثورةً في مختلف جوانب الحياة اليومية. وَمع ذلك، يُثير هذا النُّمو غير المَسبُوق مخاوف أمان وَخصوصيَّة مُلحَّة يُمكِنُ أن تُضَلِّلَ فوائد IoT بشكل كبير. تأمين هذه الأنظمة يثير تحديات متعددة الجوانب، بشكل خاص فيما يتعلق بإدارة المفاتيح في بيئات محدودة الموارد وَواسعة الانتشار. نسعى من خلال أطروحَة الدكتوراه هذه إلى مُعالَجَة التَّحدّيات الأمنيَّة وَمشاكِل الخصوصية الكامنة في IoT على مستويات مُتَعَدِّدَة.

قدَّمنا اقتراحين لِحُلول أمنيَّة قويَّة وَفعَّالة مُصَمَّمَة للتَّصدّي للتَّهديدات النَّاشِئَة من عوالم الإنترنت. المُساهمة الأولى تُقدِّمُ حلًّا مُبتَكَرًا لنظام لإدارة المفاتيح من خلال استخدامها لِلمُتَجَهَّات المُوزَّعة مُسبقًا لِلتَّغلبِ على القيود التي تواجِهُها الحلول السَّابِقة، ممَّا يُعزِّزُ الأمان لِعمليات إنشاء المفاتيح، تجديدها، وَإلغاءها. نقوم من خلال هذا الحلّ أيضًا بتقسيم المنطقة المُخَصَّصَة لِلشَّبكة إلى مُناطِق فرعيَّة، ممَّا يَجعل الحل الخاص بنا مَرِنًا، قابلًا للتوسيع، وَمُقاومًا لِلعديد من الهجمات. كَما أنَّهُ بالإضافة إلى ذلك، يُحقِّق تقليلاً كبيرًا في مُتَطَلَّبات التَّخزين، حيث يحقِّق مكاسب جِدّ كبيرة وَمُقنعة من حيثُ التَّحقُّقِ من صِحَّتِه من خلال استخدام منطِقِ التَّحليل BAN (Burrows Abadi Needham) . يَتَمَتَّعُ هذا الحلُّ بالكفاءة مقارنة بالحلول الأخرى. حَيثُ أنَّهُ يُحقِّق تقليلاً كبيرًا في متطلبات التَّخزين، مُوفرًا أكثرَ من 81.84٪ من حيث الاتصالات، خلال عملية إنشاء المِفتاح الجماعي، يُحقِّق كفاءة بنسبة 100٪. علاوة على ذلك، يَتِم تقليل العِبء الحسابي بشكل كبير من خلال تقليل عدد عمليات الضَّرب بنسبة 50٪، وَتحسين استهلاك الطاقة، ممَّا يُوَفِّر ما يصل إلى 99.99٪ خلال مرحلة إنشاء المِفتاح الجماعي. بالإضافة إلى ذلك، تُظهِر النَّتائِجُ أنَّ هذا الحَلَّ يتمَتَّعُ بمقاومَةٍ أكبر ضِدَّ هجمات الاستيلاء على العُقدة بنسبةٍ تصل إلى 96.43٪ خلال مرحلة التَّهيئَة وَبنسبةٍ تزيد عن 9.89٪ بعد ذلك، ممَّا يجعله مناسبًا لاستخدامه في شبكات IoT المحدودة الموارد.

المُساهمة الثَّانيَّة، تُركِّزُ بشكلٍ خاص على عمليَّة إلغاء المفاتيح، حيثُ تُقدِّمُ حلًّا مُبتكرًا يَستَخدِم تكنولوجيا البلوكشين وَالعقود الذَّكيَّة لِمُعالَجَة تحديات الكفاءة وَالأمان المرتبطة بالطُّرُق التَّقليديَّة لإلغاء المفاتيح.

نتائج هذه المساهمة مذهلة بالمثل: تقليلُ العِبءِ الاتصاليّ بنسبةِ 93.55٪، 91.87٪، وَ 99.75٪ مقارنةً بالحلول الأخرى خلال حالات الاستيلاء، وَالمغادرة، وَنفاذ الطَّاقة على التَّوالي، وَهذا يثبت فعاليةَ وَملائمَةَ الحَلِّ لِشبكات IoT. .

الكلمات المفتاحيَّة: إنترنت الأشياء (IoT)، شبكات الاستشعار اللاسلكية (WSN)، الحماية، إدارة المفاتيح، الأجهزة المحدودة الموارد، تكنولوجيا البلوكشين، العقود الذكية.

Acknowledgements

In the name of ALLAH the most gracious and the most merciful.
I thank ALLAH for giving me the strength and the will to accomplish this work.

I would like to express my deepest gratitude to everyone who has contributed to the successful completion of this doctoral thesis. Their support and encouragement have been invaluable throughout this journey.

First and foremost, I am indebted to my thesis advisor and co-advisor: Pr. Sofiane Mounine Hemam and Dr. Mohamed-Lamine Messai, whose guidance, expertise, and unwavering support have been instrumental in shaping the direction of this research. Your mentorship and dedication to the field have been truly inspiring.

Furthermore, I want to thank my family for their unwavering support and encouragement throughout my academic journey. Your love and understanding have been my greatest source of strength.

Lastly, I extend my gratitude to the entire academic community in the field of computer science. The collective efforts of researchers worldwide have paved the way for this work and continue to drive the pursuit of knowledge.

In conclusion, the completion of this thesis would not have been possible without the support and collaboration of everyone mentioned above. I am sincerely thankful for their contributions to my academic and professional growth.

Dedication

To the ceaseless journey of intellectual exploration and the relentless pursuit of knowledge,

To the invaluable hours devoted to inquiry and research,

To the rigorous challenges that have catalyzed my academic growth,

To the steadfast encouragement of my mentors and loved ones,

To the resilience and tenacity that have guided me to this juncture,

To all these influences and beyond, this thesis is respectfully dedicated.

In fond memory of my family members, whose enduring faith in my abilities fuels my ambitions.

To my father, whose insightful wisdom and counsel have been my compass.

To my mother, whose love and sacrifice form the bedrock of my endeavors.

To my brothers, whose support and camaraderie have been constant.

To my sisters, whose affection and backing have enriched my journey.

To my wife, whose consistent love and support serve as a pillar of strength.

To my family at large, whose unconditional love and inspiration guide my way.

To my mentors and friends, whose insightful guidance and fellowship have been invaluable.

To scholars and researchers in the field of computer science, whose pioneering work forms the foundation upon which this thesis stands.

May this dedication serve as a humble tribute to the spirit of curiosity and the scholarly pursuit that unites us all.

Contents

List of figures

List of tables

List of abbreviations

General introduction

The rapid expansion of the Internet of Things (IoT) has led to a profound transformation in the way we interact with technology. From smart homes to industrial automation and healthcare applications, IoT has seamlessly integrated into our daily lives, offering unprecedented convenience and efficiency. However, this exponential growth in the number of interconnected devices has ushered in an era of heightened security and privacy concerns, posing significant challenges to the stability and integrity of the IoT ecosystem. The management of cryptographic keys within resource-constrained devices emerges as a paramount concern, echoing the fundamental issues addressed in this doctoral thesis.

Resource-constrained devices, which constitute the backbone of IoT and Wireless Sensor Network (WSN), grapple with intrinsic limitations. These devices are characterized by constrained computational capabilities, limited storage capacity, and stringent energy efficiency requirements. These constraints render them ill-suited for executing complex computational and communication operations, exacerbating the challenges associated with secure key management.

In the context of IoT and WSNs, the importance of robust key management cannot be overstated. Effective key management is a linchpin in securing communication, data, and devices in these networks. The dynamic and heterogeneous nature of IoT networks amplifies the complexity of key management, demanding innovative solutions to ensure secure and efficient communication.

The first major contribution of this thesis, a robust and Efficient Vector-based Key Management Scheme for IoT networks (EVKMS), addresses a critical problem - the imbalance of overheads within key management schemes. This imbalance often leads to inefficiencies, favoring one aspect of key management at the expense of others. EVKMS introduces a novel approach that rectifies this imbalance, striving for

equitable performance across all facets of key management, a pivotal contribution to the IoT security landscape.

In historical context, conventional systems have traditionally relied upon centralized authorities to oversee and distribute cryptographic keys, a practice that has proven vulnerable to various security threats. These centralized systems introduce the precarious possibility of single points of failure and impose substantial communication overhead. The compromise or unavailability of the central authority can trigger a cascade of security breaches. For this, a second significant contribution, Blockchain-based Key Revocation using Smart Contracts for IoT networks (BKRSC-IoT), addresses the need for decentralization in key management. Leveraging the capabilities of blockchain and smart contracts, BKRSC-IoT aims to revolutionize the key revocation process. This innovative approach offers a promising avenue to enhance security, reduce communication overhead, and optimize energy consumption in IoT networks.

In conclusion, this doctoral thesis embarks on a comprehensive exploration of the multifaceted challenges posed by IoT, emphasizing the pivotal role of resource-constrained devices and the imperative of efficient key management. Through the introduction of EVKMS and BKRSC-IoT, this research seeks to contribute valuable insights and transformative solutions, fortifying the security and efficiency of communication within the dynamic realm of the IoT.

Thesis structure

This thesis is structured into several chapters, each of which contributes to a comprehensive understanding of the IoT security landscape and presents novel contributions in the form of key management solutions. The following is an overview of the chapters and their respective content:

Chapter 1: Internet of Things Overview

In this introductory chapter, we provide a comprehensive overview of the IoT, its evolution, and its significance in today's interconnected world. We discuss key concepts, trends, and the broader IoT architecture, setting the stage for a deeper exploration

of IoT security in the subsequent chapters.

Chapter 2: Literature Review

This chapter reviews existing literature on IoT security challenges and key management schemes. We delve into various approaches, including asymmetric and symmetric schemes, providing an in-depth analysis of each. This chapter sets the stage for our contributions by highlighting gaps in the current state of research.

Chapter 3: Contribution: EVKMS

This chapter presents our first major contribution, EVKMS. We start with an introduction to the network model and assumptions, followed by a detailed description of the solution steps, including initialization, pairwise and groupwise key establishment, and more. We evaluate the scheme's security and performance aspects. We also compare it to existing solutions and present experimental results.

Chapter 4: Contribution: BKRSC-IoT

In this chapter, we introduce our second significant contribution, BKRSC-IoT. We provide preliminary information, notations, and assumptions, followed by an explanation of the solution steps. We also perform a comprehensive evaluation, comparing it with existing solutions and presenting experimental results.

Chapter 5: Conclusion and Future Perspectives

The final chapter summarizes the key findings of this thesis and their implications for IoT security. We discuss the limitations of our work and propose potential directions for future research in this field.

Chapter 1

Internet of Things Overview

1.1 Introduction

In recent years, the Internet of Things (IoT) has garnered substantial attention, offering the promise of profound advantages for humanity. This transformative concept, initially conceived by Kevin Ashton in 1999, envisions a realm where connectivity transcends boundaries, linking entities ubiquitously and perpetually [30]. At its core, the IoT imbues "things" with the capacity for sensing, processing, and actuation, enabling these entities to collaborate autonomously, ushering in a new era of intelligent and innovative services.

The IoT's ambit is vast, encompassing a diverse array of application domains, spanning from the automation of homes to environmental monitoring and healthcare [62]. It is, in essence, a unifying force that harmonizes these disparate domains, coalescing them under the overarching canopy of what is commonly referred to as "smart life" [62]. This convergence has far-reaching implications, as it paves the way for a holistic and interconnected ecosystem where technology augments the human experience.

At its core, the IoT architecture is designed to accommodate a multitude of het-

erogeneous devices, each equipped with unique capabilities, and integrates an array of communication technologies. These technologies form the backbone of IoT, facilitating the seamless connectivity of devices, thereby enabling them to deliver essential services to end-users. This architectural framework is the linchpin of the IoT's functionality, and understanding its intricate facets is paramount to comprehending the IoT's significance.

This chapter serves as an intellectual cornerstone, providing a comprehensive overview of fundamental IoT concepts. It commences with a meticulous exploration of IoT's definitions, shedding light on the varied interpretations that underpin this paradigm. These definitions serve as the bedrock upon which our understanding of the IoT is built.

Moreover, our academic journey extends beyond definitions to encompass the breadth of IoT's potential applications. From the automation of daily life to the monitoring of our environment, the IoT offers a panoply of possibilities, each with its unique implications for society.

As we delve deeper into this chapter, we unveil the intricate architecture of the IoT. This architectural framework constitutes a complex interplay of elements and protocols that orchestrate the operations of the IoT. These protocols and components are the underpinnings of the IoT's functionality, allowing devices to communicate, interact, and deliver services seamlessly.

Through this academic exploration, we aim to equip ourselves with the foundational knowledge necessary to navigate the complexities of the IoT landscape. As we peel back the layers of this transformative concept, we prepare ourselves to comprehend its multifaceted impact on the ever-evolving field of computer science.

1.2 Definitions

The Internet of Things concept has a well-established history, with diverse interpretations and definitions. As articulated by the International Telecommunication Union (ITU) in 2012 [29], IoT is characterized as a global infrastructure facilitating the information society's advancement. It achieves this by interconnecting physical and virtual entities through evolving and interoperable information and communica-

tion technologies. IoT embodies the vision of ubiquitous connectivity, enabling the interaction of people and objects at any time, in any place, via any means, and with any service, as ideally summarized by Perera et al. [51].

Beyond mere machine-to-machine (M2M) communication, IoT is envisioned to offer an extensive array of connectivity across devices, systems, and services, spanning various protocols, domains, and applications [59]. This interconnected network of embedded devices, often referred to as "smart objects," holds the potential to revolutionize automation across diverse domains, including the realization of a Smart Grid, as elucidated by Parashar et al. [50]. The term "things" within the IoT lexicon encompasses a broad spectrum of devices, ranging from medical implants for heart monitoring to biochip transponders for livestock tracking, as well as sensors embedded in coastal waters, connected vehicles, and field operation equipment aiding firefighters in rescue missions.

Current market instances underscore the IoT's practicality, including the adoption of smart thermostat systems [49] and household appliances like washers and dryers equipped with Wi-Fi connectivity for remote monitoring. This pervasive connectivity, however, begets an exponential surge in data generation from diverse geographical locations. Consequently, there is an escalating demand for enhanced mechanisms to index, store, and process this deluge of data [52].

1.3 IoT Architecture

The IoT has a layered architecture, as demonstrated in Figure 1.1. It consists of three layers namely, the perception layer, the network layer, and the application layer [39].

1.3.1 Perception Layer

The perception layer is the lowest layer of the IoT architecture. It is responsible for collecting data from the real world. wireless sensors and actuators are the main components of this layer.

- Wireless sensors: They are responsible for collecting data from the real world.

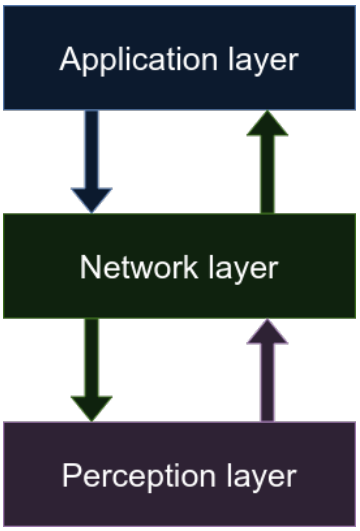


Figure 1.1: IoT architecture

They play a crucial role in the IoT networks as they are the primary data source that serves the upper layers by sending the collected data to them. In most cases, they are equipped with a microcontroller, a wireless communication module, and a power source. They are characterized by limited resources regarding storage, energy, and computational capabilities.

- Actuators: They are responsible for performing actions in the real world.

1.3.2 Network Layer

Situated at the intermediary level of the IoT architectural stack, the network layer primarily functions to facilitate communication between the perception layer and the application layer. Predominant technologies deployed at this stratum encompass Bluetooth Low Energy (BLE), Zigbee, Long Range Wide Area Networks (LoRaWAN), and IPv6 over Low power Wireless Personal Area Networks (6LoWPAN).

- **BLE**: It is a wireless personal area network technology that is used for short-range communications. This low-energy technology finds extensive application in numerous IoT contexts, including but not limited to smart residential environments, intelligent urban infrastructures, and healthcare systems. The main difference between BLE and Bluetooth is that BLE is designed for low power consumption [17] and low data rate applications where it does not require a lot

of power to operate which is beneficial for IoT applications.

- **Zigbee**: It is another wireless communication protocol that is suitable for low-power and low-data rate applications. It is based on the IEEE 802.15.4 standard [28]. In IoT, it is used in many applications such as home automation, industrial control, and healthcare because of its energy efficiency, reliability, and scalability.
  Mesh networking [2] is supported in Zigbee, where devices can form self-organized networks. In a Zigbee mesh, each device can relay data to other devices, increasing network reliability and range as mentioned before. If one of the devices becomes unreachable, there will be a path on which data can still find its way to the intended destination through other devices in the mesh.
  From another perspective, Zigbee is not suitable for applications that require high bandwidth like video streaming.
  The organization of Zigbee has defined interoperability standards by the Zigbee Alliance [6], which is a non-profit organization responsible for defining the standards of Zigbee. These interoperability standards ensure that Zigbee devices from different vendors can communicate with each other.

- **LoRaWAN**: It is a wireless communication protocol that is suitable for long-range applications. It is based on the LoRa technology [18]. It is designed for long-range and low-power IoT and machine-to-machine (M2M) applications. It is particularly apt for scenarios requiring the transmission of modest data volumes over extended ranges while maintaining minimal energy expenditure.
  In addition to long-range, low power consumption, and low data rate, LoRaWAN has other features such as bidirectional communication, which means that devices can send and receive data; This feature enables information and data to be sent to the devices that are in the field, which is beneficial in enhancing their functionality. Another feature of LoRaWAN is scalability [5], which means that it can support network growth by adding a large number of devices.
  The topology of LoRaWAN is a star topology, where each device is connected to a gateway, and these gateways are connected to a network server on which they transmit the data they gathered from the end devices of the previous perception layer.

- **6LoWPAN**: It is a wireless communication protocol based on the IPv6 protocol [33]. It enables the sending and receiving of IPv6 packets over power-constrained wireless networks and is specifically designed for use with low-power devices and networks, which have limited power, memory, and processing re-

sources; This makes it suitable for IoT applications.

Table 1.1 shows a comparison between the most popular IoT communication technologies that are used in the network layer.

Characteristic	BLE	Zigbee	LoRaWAN	6LoWPAN
Communication Range	Short (typically <100 meters)	Short (typically <100 meters)	Long (several kilometers)	Short to Moderate
Power Consumption	Low	Low	Low	Low
Data Rate	Low to Moderate	Low to Moderate	Low	Low
Network Topology	Star	Mesh	Star-of-Stars (Gateway-based)	Mesh (Optional)
Interoperability	Yes (within BLE ecosystem)	Yes (within Zigbee ecosystem)	Yes (within LoRaWAN ecosystem)	Yes (IPv6-based)
Security Features	Strong	Strong	Strong	Strong
Scalability	Limited	High	High	High
Application Areas	IoT devices, wearables	Home automation, industrial control	Smart cities, agriculture	IoT devices, sensor networks
Device Complexity	Low to Moderate	Moderate to High	Low to Moderate	Low to Moderate
Use Cases	Healthcare, smart homes	Home automation, industrial control	Smart agriculture, smart cities	Environmental monitoring, healthcare
Key Standardization Bodies	Bluetooth SIG	Zigbee Alliance	LoRa Alliance	IETF (Internet Engineering Task Force)
Key Advantages	Low power, wide device support	Mesh networking, interoperability	Long-range, energy-efficient	IPv6 integration, efficient header compression
Key Challenges	Limited range, device diversity	Complex network setup, limited range	Limited data rate, gateway dependence	Limited range, header compression overhead

Table 1.1: Comparative analysis of prominent IoT communication technologies in the network layer

1.3.3 Application Layer

Positioned at the apex of the IoT architectural hierarchy, the application layer constitutes the uppermost level. It is responsible for enabling services and applications that are built on top of the network layer. It is the layer that is visible to the end-user. In this layer, the gathered data from the perception layer is analyzed and processed to extract useful information; This information is then used to make decisions and take actions.

Key features of the IoT application layer

The IoT application layer serves as the interface between the end-user and the IoT network. It is considered the central hub for multiple functions that make IoT systems useful and drive their functionality as well. This layer is responsible for:

- **Data processing and analysis:** As mentioned before, the data that is gathered from sensors and actuators from the environment, is analyzed and processed in this layer to transform them into meaningful insights.
- **Business Logic and Decision Making:** Within this layer, the decision-making and business logic processes are implemented by creating rules, algorithms, and logic to interpret data and determine appropriate actions. For example, in smart farms, the application layer may decide to adjust water irrigation based on temperature readings.
- **Application Services:** IoT application services are developed and deployed in this layer. These services provide specific functionalities or applications to end-users or systems, catering to diverse domains like home automation, industrial control, smart agriculture, and environmental monitoring.
- **Data Storage and Database Management:** The application layer often includes databases and data storage solutions to retain historical data collected from IoT devices. This historical data is valuable for trend analysis, reporting, and long-term decision-making.
- **User Interfaces:** User interfaces, including web dashboards, mobile apps, and desktop applications, are commonly integrated into the application layer.

These interfaces empower users to interact with and manage IoT devices, access real-time data, configure settings, and view insights and reports.

- **Integration with External Systems:** IoT applications are often integrated with external systems, such as cloud services, third-party Application Programming Interface(s) (API(s)), or enterprise systems. The application layer facilitates the data exchange and the collaboration with these external entities.
- **Scalability:** Given the potential scale of IoT deployments, the application layer must be designed with scalability in mind. It should handle increasing data volume and user demands efficiently.
- **Security:** Security measures, such as authentication, encryption, and access control, are a paramount concern in the application layer. These safeguards protect data and IoT devices from cyber threats, ensuring the privacy and integrity of the system.
- **Real-Time Processing:** Some IoT applications require real-time data processing to trigger immediate actions or alerts. The application layer must have real-time processing capabilities to meet these requirements, enabling timely responses to critical events and situations.
- **Protocols and Communication:** IoT applications rely on various communication protocols to facilitate data exchange between devices and the application layer. Common protocols like MQTT, CoAP, and HTTP are employed to ensure effective communication. In the following points, we discuss these protocols in more detail:
  - **MQTT:** is a lightweight publish/subscribe messaging protocol designed for devices with limited resources and networks that are either high-latency, low-bandwidth or unreliable. It is an open standard protocol that is used in many IoT applications. It is based on the publish/subscribe model, where publishers publish messages to a broker, and subscribers subscribe to topics on the broker to receive messages. It is a lightweight protocol that is suitable for IoT applications because it does not require a lot of power to operate, which is beneficial for IoT applications. It is also a reliable protocol because it guarantees that messages will be delivered to the subscribers. It is also a secure protocol because it supports authentication and encryption.
  - **CoAP:** CoAP is a protocol situated within the application layer, specifically engineered to accommodate the stringent resource constraints of both

devices and networks with limited capabilities. Its foundational principles draw heavily from the REST architectural style, as expounded in [54]. It is a lightweight protocol that is suitable for IoT applications because of its low overhead. It supports encryption and authentication, which makes it a secure protocol. Its predefined message types are "reset", "confirmable", "non-confirmable", and "acknowledgment". Some of the features of CoAP that the first versions of HTTP did not support are push notifications, where the server can send a message to the device without the device requesting it, in addition to resource discovery, involving the server's storage of a list detailing network-available devices and resources.

– **HTTP:** is a protocol that is used to transfer data between a client and a server. It is a reliable protocol because it guarantees that the data will be delivered to the client. It is based on the request/response model, wherein the client transmits a request to the server, and in turn, the server replies to the client. The underlying foundation is the TCP protocol, a connection-oriented framework that necessitates a pre-established link between client and server prior to data exchange. HTTPS (HTTP Secure) is a secure version of HTTP that employs SSL/TLS to encrypt the data that is transferred between the client and the server, making it a secure protocol.
Table 1.2 shows a comparison between MQTT, CoAP, and HTTP.

1.4 Conclusion

In this chapter, we have covered essential aspects of the IoT landscape. We initiated our discussion with a definition of the IoT concept, followed by an in-depth exploration of its multi-tiered architecture, including the perception, the network, and the application layers. A particular emphasis was placed on elucidating the key features of the IoT application layer.

With a solid foundation in IoT fundamentals, our academic journey now leads us to a critical juncture. In the next chapter, we will embark on a comprehensive literature review of key management in IoT security. This review will offer valuable insights into the existing landscape of key management strategies within IoT security, identifying trends, challenges, and advancements. As we delve deeper into this

Characteristic	**MQTT**	**CoAP**	**HTTP**
Protocol Type	Publish-Subscribe	Request-Response	Request-Response
Lightweight	Yes	Yes	No
Message Reliability	Configurable (QoS levels 0-2)	Configurable (Confirmable and Non-confirmable)	Reliable (TCP-based)
Resource-Oriented	No	Yes	Yes
Security	Yes	Yes	Yes (Supports HTTPS - TLS/SSL)
Typical Use Cases	M2M communication, IoT with constrained devices	IoT with constrained devices, resource discovery	Web-based applications, IoT with high-speed networks
REST	No	Yes	Yes

Table 1.2: Comparative study of MQTT, CoAP, and HTTP

specialized domain, we lay the groundwork for the contributions and research that will follow, enriching our understanding of IoT security and its critical components.

Chapter 2

Literature Review

2.1 Introduction

The present chapter embarks on an extensive exploration of the landscape surrounding the security challenges encountered within the realm of the IoT. As discussed in the preceding chapter, "Chapter 1: Internet of Things Overview," the IoT has witnessed exponential growth, revolutionizing the way devices interact and communicate. However, this rapid proliferation has brought forth a multitude of security concerns that demand rigorous examination and comprehensive solutions.

The objective of this literature review is to dissect and evaluate the existing body of knowledge regarding IoT security challenges and, more specifically, the various key management schemes proposed to mitigate these challenges. Key management is undeniably a linchpin in the broader scope of IoT security, as it directly impacts the confidentiality, integrity, and availability of data transmitted and stored within the IoT ecosystem.

Building upon the foundational knowledge presented in Chapter 1, this chapter delves deeper into the intricacies of IoT security. It begins by identifying and categorizing the overarching security challenges that plague IoT deployments. Sub-

sequently, it conducts an exhaustive survey of existing key management schemes, further subdivided into asymmetric and symmetric schemes, each playing a pivotal role in securing IoT communications.

In the following sections of this chapter, we conduct an in-depth analysis of these key management schemes, scrutinizing their applicability, strengths, and weaknesses. This analysis spans a wide spectrum of approaches, ranging from matrix-based methodologies to polynomial-based solutions, each tailored to address specific IoT scenarios and constraints. Furthermore, we examine the trade-offs between security and performance, as well as deployment strategies for heterogeneous IoT environments.

Ultimately, this literature review seeks to provide a comprehensive foundation for the subsequent chapters, which will delve into the design and evaluation of novel key management schemes. By synthesizing existing knowledge while critically assessing the nuances of each approach, this chapter strives to contribute to the evolving discourse on IoT security, propelling us towards more resilient and secure IoT ecosystems.

2.2 IoT Security Challenges

IoT devices have come to the forefront as a disruptive influence in the modern digital landscape, enabling a multitude of applications, ranging from smart homes to industrial automation. However, amidst the rapid proliferation of these devices, one of the most pressing challenges that researchers and practitioners alike must grapple with is the inherent resource and power limitations that define IoT ecosystems. This challenge is pivotal in the realm of IoT security, where the ability to ensure the confidentiality and integrity of data and communications hinges on the device's computational capabilities, memory, and energy supply.

Resource-constrained IoT devices, by their very nature, are characterized by limited computational resources, often featuring low-power processors and modest memory capacities. These devices are designed to perform specific tasks efficiently while consuming as little energy as possible. While this design philosophy makes IoT devices suitable for their intended applications, it simultaneously renders them vulnerable to security threats. Traditional cryptographic algorithms, known for their computational complexity, may strain the processing capabilities of these devices, making

them susceptible to attacks that exploit computational weaknesses.

Furthermore, the constraint on memory capacity can hinder the storage of cryptographic keys and security parameters. In many IoT scenarios, especially those involving remote and unattended deployments, the ability to securely manage keys and credentials is paramount. The scarcity of memory resources can limit the device's ability to store cryptographic material securely, potentially exposing sensitive data to unauthorized access.

Another facet of this challenge lies in the limited energy supply of IoT devices, often relying on battery power. These devices are expected to operate for extended periods without frequent battery replacements or recharging. Security mechanisms that demand continuous, energy-intensive cryptographic operations can significantly reduce the device's operational lifespan, making energy-efficient security protocols a necessity.

In the context of IoT security, addressing these resource and power limitations is not merely a matter of convenience but a fundamental requirement. Failing to account for these constraints can lead to vulnerabilities that malicious actors can exploit to compromise the confidentiality and integrity of IoT data and communications. Therefore, any comprehensive security protocol for IoT must consider innovative strategies that balance security with resource efficiency, ensuring that security measures do not unduly burden the devices.

In the subsequent sections of this literature review, we will delve deeper into the strategies and solutions proposed by existing key management schemes in mitigating these resource and power limitations. Additionally, we will assess the extent to which these schemes have succeeded in optimizing resource utilization and energy efficiency, thereby paving the way for more secure and sustainable IoT ecosystems.

2.3 Existing Key Management Schemes

The realm of IoT security is characterized by its dynamic and ever-evolving nature, driven by the growing ubiquity of connected devices. A fundamental aspect of securing these networks lies in the efficient management of cryptographic keys, which is critical for ensuring data confidentiality and integrity. Key management schemes in

IoT networks have been proposed extensively in the literature, each offering unique approaches to address the complex challenge of securing resource-constrained devices while navigating the resource and power limitations inherent to the IoT ecosystem.

2.3.1 Asymmetric Schemes: Balancing Security and Resource Constraints

Asymmetric key management schemes have emerged as a promising avenue for achieving strong security in IoT networks. These schemes, exemplified by the [12, 1, 31, 40, 4, 10, 42, 56] works, rely on the computational complexity of mathematical problems to safeguard cryptographic keys. The robustness of these cryptographic schemes relies fundamentally on the computational challenge associated with deriving the private key from its corresponding public key.

However, the allure of strong security comes with a trade-off. Asymmetric schemes impose substantial computational and communication overheads on IoT devices, which are inherently resource-constrained. The computational complexity of solving mathematical problems can strain the limited processing power and memory capacity of IoT nodes. This strain becomes more pronounced as the complexity of the underlying mathematical problem increases. Consequently, while asymmetric schemes offer robust security, they may not be well-suited for resource-constrained IoT environments.

2.3.2 Symmetric Schemes: Pre-distribution and Post-distribution Strategies

Symmetric key management schemes provide an alternative approach to IoT security. These approaches can be categorized into two distinct subgroups: pre-distribution techniques and post-distribution strategies, each possessing its individual array of benefits and associated complexities.

Pre-distribution Schemes

In pre-distribution schemes, represented by well-known schemes like Blom's scheme [16] and its extensions by Du et al. [22] and others [67, 69, 45, 46, 47, 44], and before deployment, shared context and keys are preloaded into IoT nodes. While these schemes offer a high level of security, they face two primary challenges.

First, pre-distribution schemes demand significant storage overhead. IoT devices must store a substantial number of keys and associated data, which can strain the limited memory resources available on these devices. This storage requirement can be a limitation, particularly in scenarios where IoT nodes are severely resource-constrained.

Second, pre-distribution schemes require robust network connectivity. For relatively large IoT networks, ensuring consistent connectivity among nodes becomes a challenge. In sparsely connected networks or those with limited initial deployment infrastructure, pre-distribution schemes may encounter practical difficulties.

Post-distribution Schemes

Post-distribution schemes, as exemplified by AbuAlghanam et al.'s work [1] and Gautam et al.'s scheme [27], and others [11, 24, 58, 61, 25, 63], distribute keys after IoT device deployment. This approach mitigates some of the storage and connectivity challenges associated with pre-distribution schemes.

Nevertheless, post-distribution schemes introduce their own set of considerations. They often involve a higher communication overhead, as keys must be communicated to nodes after deployment. This increased communication may lead to potential vulnerabilities, such as interception and eavesdropping, during the key distribution process.

In the following sections, we will delve deeper into specific key management schemes, providing detailed insights into their methodologies, strengths, and limitations. These discussions aim to contribute to a comprehensive understanding of the intricate landscape of IoT key management and its direct relevance to addressing resource and power limitations, a critical aspect of IoT security.

2.3.3 In-Depth Analysis of Key Management Schemes

Blom's Scheme: An Exploration of the Matrix-Based Approach

Blom's scheme [16] stands as an early and influential example of pre-distribution key management schemes. It aims to create a matrix of keys with dimensions $N \times N$, Each element within the matrix signifies a key, correlating to two nodes identified by specific identifiers. Blom's scheme operates using three pivotal matrices: a public matrix G, a random matrix D, and a secret matrix A. G has $\lambda + 1$ rows and N columns, D has $\lambda + 1$ rows and $\lambda + 1$ columns, and A has $\lambda + 1$ rows and N columns. To facilitate the construction of a new key matrix K, typically represented as $K = AG$. The importance of symmetry is emphasized to ensure that $K[i, j] = K[j, i]$.

Upon deployment, each node broadcasts its public column $G[x, i]$ to neighboring nodes. This act precedes the computation of pairwise keys $K[i, j]$ or $K[j, i]$ between nodes i and j by multiplying the shared column $G[x, i]$ with the corresponding row $A[j, x]$, which can also be expressed as $G[x, j] * A[i, x]$. Blom's scheme introduces a threshold parameter λ, signifying the maximum number of compromised nodes the network can tolerate without compromising its overall security. If the number of compromised nodes exceeds this threshold, it becomes possible to deduce the entire keys matrix K, rendering the network insecure.

Du et al.'s Extension: Optimizing Connectivity and Resilience

Du et al. [22] built upon the foundation laid by Blom's scheme and introduced a novel extension that focuses on enhancing network connectivity and resilience against compromising attacks. This extension introduces the concept of key-spaces, where a set of random symmetric matrices, denoted as D matrices, are employed. A node may hold the entire set of D matrices or a subset of them. Communication between two nodes is only possible if they share a common key space.

The key insight behind Du et al.'s extension is to reduce unnecessary connectivity among nodes, which not only enhances network resilience but also improves communication efficiency. However, this approach comes with increased storage requirements, as each node must store an identifier and τ key spaces. This storage overhead significantly exceeds that of Blom's original scheme, which only necessitates the storage

of $\lambda + 1$ matrix elements. Additionally, the computational load increases due to the generation of public columns for nodes, with each node performing $2\lambda + 1$ multiplications and λ additions, compared to Blom's proposal, which requires only $\lambda + 1$ multiplications and λ additions.

Yu and Guan's Deployment Knowledge-Based Solution

Yu and Guan [67] introduced a key management solution based on Blom's scheme, tailored for specific deployment scenarios, particularly in IoT contexts. The authors considered dividing the deployment area into several sub-areas, with nodes grouped accordingly. During the initial key pre-distribution phase, a common global matrix G, akin to Blom's, is assigned to all nodes. Each group of nodes is assigned a unique secret matrix A. Additionally, a set of matrices B is assigned to each individual node. These matrices are preloaded onto nodes, along with their corresponding column of the G matrix, the row of the A matrix associated with their group, and a selection of rows from their assigned B matrices.

Key establishment between nodes occurs in three distinct cases: when nodes belong to the same group and share the same A matrix, when nodes do not belong to the same group but share at least one B matrix, and when nodes share neither group nor B matrices. Each of these scenarios necessitates specific calculations for pairwise key establishment.

While Yu and Guan's solution offers improved network connectivity and resilience, it also introduces storage and energy consumption challenges. The scheme requires extensive storage capacity due to the multitude of matrices that need to be preloaded onto nodes. Moreover, the increased computation required for generating public columns and performing key calculations can have implications for energy efficiency, particularly on resource-constrained IoT devices.

Zhang et al.'s Matrix-Based Approach for Heterogeneous Devices

Zhang et al. [69] proposed a matrix-based key management scheme tailored for smart homes and IoT environments featuring heterogeneous devices from various manufacturers. This scheme relies on two fundamental keys: the master secret key

and the session keys. Upon joining the network, a node extracts the master secret key, typically generated using a key generation approach, such as Wang et al.'s method [66]. This master secret key secures communications between the home gateway and the constrained node.

One notable advantage of this scheme is its energy-efficient approach. By delegating intensive operations to home gateways, the network minimizes energy consumption at the device level. However, each device is still required to perform $2\lambda + 1$ multiplications, which represents an increase compared to Blom's scheme's relatively modest requirement of $\lambda + 1$ multiplications. Additionally, communication overhead is higher compared to previous schemes, as devices need to communicate with their mesh routers to establish session keys.

Mesmoudi et al.'s SKWN Scheme

In the domain of hierarchical wireless sensor networks, the contributions of S. Mesmoudi et al. stand out, particularly as outlined in [43]. The authors propose a sophisticated key management scheme, termed SKWN, that encompasses key establishment, renewal, and the integration of new nodes. Utilizing machine learning algorithms, SKWN dynamically adjusts its security levels based on ongoing network activities. An Intrusion Sensing Algorithm (ISA) component facilitates real-time intrusion detection, obviating the need for persistently high security settings in the absence of active threats. SKWN is praised for its scalability, flexibility, and robust security architecture. Nonetheless, the scheme has a high communication overhead, reflected in the large number and size of messages exchanged during its operational phases.

Further elucidation on key revocation scenarios, where Mesmoudi et al. outline two primary cases based on node types: the Cluster Head (CH) and the Cluster Member (CM).

- **Cluster Head compromise:** When a compromised CH is identified by the Base Station (BS), key revocation is initiated for both the CH and its associated CMs, followed by a key renewal process for the remaining nodes.
- **Cluster Member compromise:** If a CH identifies a compromised CM, the

key of the compromised CM is revoked. The other CMs in the cluster are notified, and a key renewal process is initiated.

However, there are key limitations that warrant further investigation:

- **Leaving or draining scenarios:** The scheme does not consider key revocation for nodes that exit the network or are incapacitated due to battery drainage. This presents a gap in the protocol that could have significant implications for real-world deployment.
- **Communication overhead:** As previously noted, the large number and size of messages could be a concern for resource-constrained sensor networks.
- **Dynamic security level:** The adaptability of the security level, while innovative, may expose the network to vulnerabilities during periods of lower security.
- **Intrusion detection efficiency:** The computational cost and effectiveness of the ISA component for real-time intrusion detection have not been detailed, posing questions about its scalability and reliability.

Nafi et al.'s Matrix-Based Scheme with Security Trade-offs

In a study by Nafi et al. [45], another matrix-based approach was presented, influenced by the foundational work of Blom's scheme. The design involves the predistribution of an $N \times N$ matrix, which does not need to be symmetric. It is characterized by the removal of the initial materials after the deployment and discovery phases. To calculate shared session keys between arbitrary pairs of nodes, the scheme employs determinant calculations for square matrices of dimensions 2×2 that contain the elements of the intersection of their rows and columns. The authors also proposed other different phases, starting from node addition to key refresh.

While offering high connectivity and security during the pairwise key establishment phase, this scheme exhibits a trade-off in security during the new node addition phase. During this phase, group-wise keys are communicated in plaintext through a public channel, introducing potential vulnerabilities. Additionally, the scheme requires substantial storage capacity, as each node must store an $N \times N$ matrix, which can be a challenge for resource-constrained IoT devices in large-scale networks.

The key revocation process in the described scheme operates under two different scenarios, each necessitating specific actions to maintain the security and integrity of the network.

- **Case 1: Malicious activity detected by a gateway node**

  In this instance, the revocation process is initiated by a gateway node detecting anomalous behavior from an IoT device, indicative of a security threat. The following steps are executed:

  - **Key deletion:** The gateway node removes the pairwise key associated with the identified malicious node.
  - **Matrix update:** The corresponding row and column for the malicious node in the neighbors' matrix are removed. Additionally, the node's id is deleted from the neighbors' vector.
  - **Group key re-establishment:** A new group-wise key is generated by rerunning the relevant algorithm.
  - **Notification:** The gateway node disseminates a message to neighboring nodes containing the malicious node's id, a nonce, and the newly generated group key.
  - **Acknowledgment and cleanup:** Receiving nodes authenticate the received message, remove any data linked with the malicious node, and send an acknowledgment back to the gateway node.

- **Case 2: Voluntary exit from the network**

  In contrast, this scenario involves an IoT device voluntarily exiting the network. The node undergoes the following steps:

  - **Leave notification:** The leaving node sends a $LEAVE$ message to its one-hop neighbors, including its id and a nonce.
  - **Cleanup:** Upon receiving this message, neighbor nodes erase all relevant data associated with the leaving node.
  - **Acknowledgment:** An acknowledgment message is sent back to the leaving node by each neighbor.
  - **Memory drain and exit:** The leaving node wipes its memory after receiving acknowledgments from all its neighbors and subsequently departs from the network.

A. K. Gautam et al.'s Polynomial-Based Scheme

A. K. Gautam et al. [27] introduced a polynomial-based scheme, primarily utilizing Lucas polynomials based on the Fibonacci series. This scheme is designed for clustered IoT networks, where cluster heads play a pivotal role in generating polynomials and random numbers.

One challenge of this scheme lies in the random selection of cluster heads. This randomness can lead to substantial energy consumption, particularly when a resource-limited node is selected as the cluster leader. The scheme exhibits resilience in some aspects but raises concerns related to energy efficiency.

Nafi et al.'s Bivariate Polynomial-Based Scheme

Nafi et al. [46] proposed a scheme based on bivariate $2n$-degree polynomials. Each node is preloaded with its corresponding polynomial, which includes the network key K_n. The scheme distinguishes between direct and indirect key establishment mechanisms, providing secure communication options for both scenarios.

However, this scheme relies on substantial preloading, including the node's memory with the polynomial's general form and the network key. It exhibits good connectivity but introduces challenges related to storage and computation overheads.

The researchers introduced a stage dedicated to removing nodes identified as compromised or departing. During this phase, nodes adjacent to the node in question are instructed to purge it from their local adjacency lists. Concurrently, they are also directed to modify their existing polynomial functions by reducing them by a single term for each leaving node. Furthermore, these adjacent nodes are tasked with erasing cryptographic keys that were previously shared with the compromised or departing node. However, the proposal falls short in explaining the communication methods needed for alerting adjacent nodes about a compromised or leaving node. Also, the authors don't specify how one would identify a compromised node.

AbuAlghanam et al.'s Hybrid Approach for Smart Cities

In their work, AbuAlghanam et al. [1] presented H2KD, a layered framework that blends asymmetric elliptic curve methods with symmetric cryptographic techniques. The architecture of this scheme involves phases for pre-deployment of keys, initialization, establishment, and updates. It distinguishes itself by its capacity for heterogeneity, scalability, and defense against attacks involving node capture.

Nevertheless, a significant shortcoming is the absence of a secure mechanism for key updates; new keys are transmitted in an unencrypted format during the updating stage.

Bai et al.'s Rabin Cryptosystem for Smart Homes

Bai et al. [12] proposed an asymmetric scheme designed for smart home applications based on the Rabin cryptosystem. This system relies on the difficulty of factoring large integers, similar to the RSA algorithm but using modular squares and modular root squares.

While offering cryptographic strength, this scheme introduces vulnerabilities related to number factorization attacks and lacks an evaluation regarding energy consumption.

Nafi et al.'s Pairwise Key Management with One-One Functions

Nafi et al. [47] presented a pairwise key management scheme that employs one-one functions and their inverses to calculate pairwise keys. This scheme involves several phases, including key preloading, addition, revocation, key refresh, and key establishment.

The scheme excels in connectivity and resilience against node capture attacks but introduces concerns related to storage requirements, energy consumption, and susceptibility to brute-force attacks, dependent on the chosen one-one functions.

In this scheme, key revocation is essentially a decentralized process driven by local

observations made by neighboring nodes. This approach can be delineated through its critical features:

- **Passive key revocation by neighbors**

  In contrast to active message-driven revocation, this scheme relies on passive monitoring. Each node is responsible for unilaterally revoking the keys of neighboring nodes that have not communicated within a specified time interval T_r.

- **Periodic key regeneration**

  The scheme mandates periodic key regeneration, which occurs every T_r time unit. This adds a layer of dynamicity to the key management, reducing the window of opportunity for an attacker to misuse a compromised key.

- **Server-driven updates**

  The server periodically sends updates in the form of new one-to-one functions to all nodes, occurring every $2 \times T_r$ time unit. This central role of the server supplements the decentralized aspect of key management in the network.

- **Attack-triggered refresh**

  In the case of an identified attack, the server proactively sends a "refresh" message via the gateway node to update all nodes. This message carries the number of detected attacks to recalculate and potentially update the value of T_r

- **Communication overhead**

  Upon receipt of a refresh message, nodes must reply to acknowledge, effectively mirroring the communication overhead found in [45].

Kandi et al.'s Decentralized Blockchain-based Scheme

In this work, attention is directed towards a blockchain-based key management scheme as posited by the authors in [34]. Unlike traditional centralized schemes, this decentralized model aims to address the single point of failure problem that plagues centralized systems. The scheme is characterized by a two-layered architecture where one layer is dedicated to key management and the other is responsible for managing the keys of nodes.

Mechanics of Key Revocation: The scheme employs unicast and broadcast methods for revoking the key of a departing node. Specifically, a unicast refresh message containing the departing node id and a randomly generated key K_R is sent to the affected node set. This is followed by a broadcast refresh message to other sets, ensuring the updation of keys among the remaining nodes.

Blockchain Participants and Verification: The scheme incorporates Blockchain Participants (BPs) who are notified upon a node's departure. They then create a new transaction containing the leaving node id and additional data. This transaction is verified by all BPs through signature and hash checks, as well as by confirming the membership status of the departing node in the received information.

Strengths and Weaknesses:

- **Decentralization:** The scheme successfully decentralizes the key revocation process, eliminating the central point of vulnerability present in traditional methods.
- **Transaction Verification:** The use of blockchain for transaction verification adds an additional layer of security, making the scheme robust against malicious activities.
- **High Communication Overhead:** Despite its merits, the scheme suffers from high communication overhead, largely due to the need for every constrained node to participate in the key revocation process. The increased communication overhead remains a challenge that may restrict its practical applicability, especially in networks with constrained resources.

Msolli et al.'s Hash-Based Scheme with Path Establishment

Msolli et al. [44] introduced a hash-based scheme based on Eschenauer and Gligor's probabilistic approach. This scheme utilizes Shift-AES as an encryption algorithm and encompasses three key phases: pre-distribution, key discovery, and path establishment.

While this scheme offers connectivity and resilience, it faces challenges related to

communication and computation overheads, particularly as the number of intermediate nodes increases during path establishment.

2.4 Conclusion

In this chapter, we have provided a comprehensive overview of the existing literature on the security challenges of IoT and key management schemes. The examination of various key management approaches, both asymmetric and symmetric, has shed light on the complexities and trade-offs involved in securing IoT networks.

Our analysis of specific schemes, such as Blom's Matrix-Based Approach, Du et al.'s Connectivity Optimization, and other notable contributions, has highlighted the diverse strategies employed to address security concerns in IoT environments. This literature review has served as a foundation for our research, providing valuable insights into the state-of-the-art in IoT security.

Having established a solid understanding of the IoT security landscape in the preceding chapters, we now transition to our contributions in the field. In Chapter 3, we introduce EVKMS our novel approach, designed to address some of the key challenges identified in the literature. Building upon the insights gained from the literature review, we present our research findings, methodologies, and contributions, which aim to enhance the security and manageability of IoT networks. Chapter 3 will delve into the details of EVKMS and its application in real-world IoT scenarios, paving the way for a deeper exploration of our research outcomes in subsequent chapters.

Contribution: A robust and efficient vector-based key management scheme for IoT networks

3.1 Introduction

The concept of the IoT revolves around connecting physical objects, such as sensors and devices, to the internet. These IoT devices possess communication capabilities, enabling them to autonomously exchange data with both each other and their surrounding environment. The applications of IoT have expanded across various domains, including industries, healthcare, agriculture, transportation, and energy management [9, 55, 7]. For instance, IoT devices play a pivotal role in remotely monitoring patient health, optimizing agricultural practices, enhancing transportation safety, and reducing energy consumption in smart buildings.

In the dynamic landscape of IoT networks, secure communication is paramount, especially considering the sensitive nature of data exchange between IoT devices. Securing communication within IoT networks is a complex challenge due to the susceptibility of wireless channels and the inherent resource constraints of IoT devices. These devices often operate with limited resources, adding to the intricacy of key management processes, including key establishment, distribution, revocation,

renewal, and addition. Therefore, the development of efficient and robust key management solutions is imperative to ensure the security of communications within IoT networks.

WSN represent a subset of IoT networks, Consisting of small, energy-limited, and cost-effective sensor nodes, these networks are designated for the collection of environmental data and wireless communication. WSNs have been instrumental in shaping key management strategies within IoT networks. Many initial approaches for securing communications and managing cryptographic keys were originally designed for WSNs. Despite the advent of a more expansive IoT ecosystem that includes a variety of internet-connected devices, WSNs continue to hold a vital role in low-power IoT scenarios. This is primarily due to their inherent attributes, such as constrained resources and energy limitations.

Notwithstanding the challenges posed by IoT networks, WSNs continue to play a vital role in the broader IoT ecosystem, particularly in low-power IoT applications. With the ongoing evolution of the IoT device landscape, there will unquestionably be a rising demand for key management schemes that are both efficient and cost-effective.

In academic literature, a variety of key management strategies have been proposed with the objective of tackling the security challenges inherent in IoT network communications. However, many of these previously discussed approaches prioritize one aspect, such as communication overhead, at the expense of others, including storage and computational costs. This imbalance results in key management schemes that excel in specific IoT network assumptions but fall short in others. Hence, there is a pressing need for a new key management scheme that meticulously balances all relevant performance criteria to ensure effective communication security within IoT networks.

This chapter presents EVKMS, a newly devised key management scheme, designed to tackle the challenges associated with securing IoT network communications. EVKMS strikes a balance between resilience and efficiency by leveraging predistributed vectors to protect stored keys and sensitive data transmitted across insecure channels. Our objective with EVKMS is to address the limitations of existing key management solutions, providing a robust and efficient approach for enhancing communication security within IoT networks.

EVKMS boasts several key features, including:

- Minimizing the size of pre-loaded data.
- Enhancing resilience against node capture attacks throughout various phases, including initialization.
- Supporting scalable networks that accommodate new node additions.
- Enabling secure establishment of pairwise and group-wise keys among IoT devices without exposing sensitive data through communication channels.
- Facilitating efficient group-wise key establishment with zero communications.
- Minimizing computational overhead.

The subsequent sections provide an outline of the structural organization within this chapter. We commence by presenting a comprehensive exposition of our proposed scheme, elucidating the network model, assumptions, and the various phases that comprise its implementation. Following this, we subject our scheme to a rigorous assessment, encompassing both formal and informal security analyses, as well as a thorough evaluation of its performance across multiple dimensions. These assessments span aspects such as data storage, computational complexity, communication overheads, and energy consumption. Finally, we conclude this chapter by summarizing our key findings and insights drawn from our research endeavors.

3.2 Network model

Our suggested approach is built on a network model that shares the same components with Nafi et al. [45] work as shown in Figure 3.1. The primary components of the model are as follows:

3.2.1 Remote server

A cloud-based server that is accessible via the Internet. In most cases, it offers API(s) for facilitating data exchange and communication.

3.2.2 Gateway nodes

These nodes possess both Internet connectivity and ample resources. Serving as intermediaries that link resource-constrained nodes to the remote server, they utilize a secure asymmetric communication protocol.

3.2.3 Limited-resource nodes

These nodes are equipped with sensors and are considered the primary data source. They have limited resources concerning storage, energy, and computational capabilities. They are connected to the gateway nodes via a wireless network.

3.3 Assumptions

We make assumptions that:

- The exchange of information between the gateway nodes and the remote server is considered secure.
- Every gateway node is granted trustworthiness and strengthened through the utilization of Trusted Platform Module (TPM).
- The remote server has no limitations or constraints regarding the availability of the resources.
- Each gateway node establishes communication with the remote server via the Internet.
- No direct Internet access is available for constrained nodes.
- Each subset is connected to either one or two neighboring subsets.
- Each subset may include one or more gateways.
- First local identifiers within subsets are always assigned to gateways.
- Security and authentication among gateways are ensured using public key cryptography.

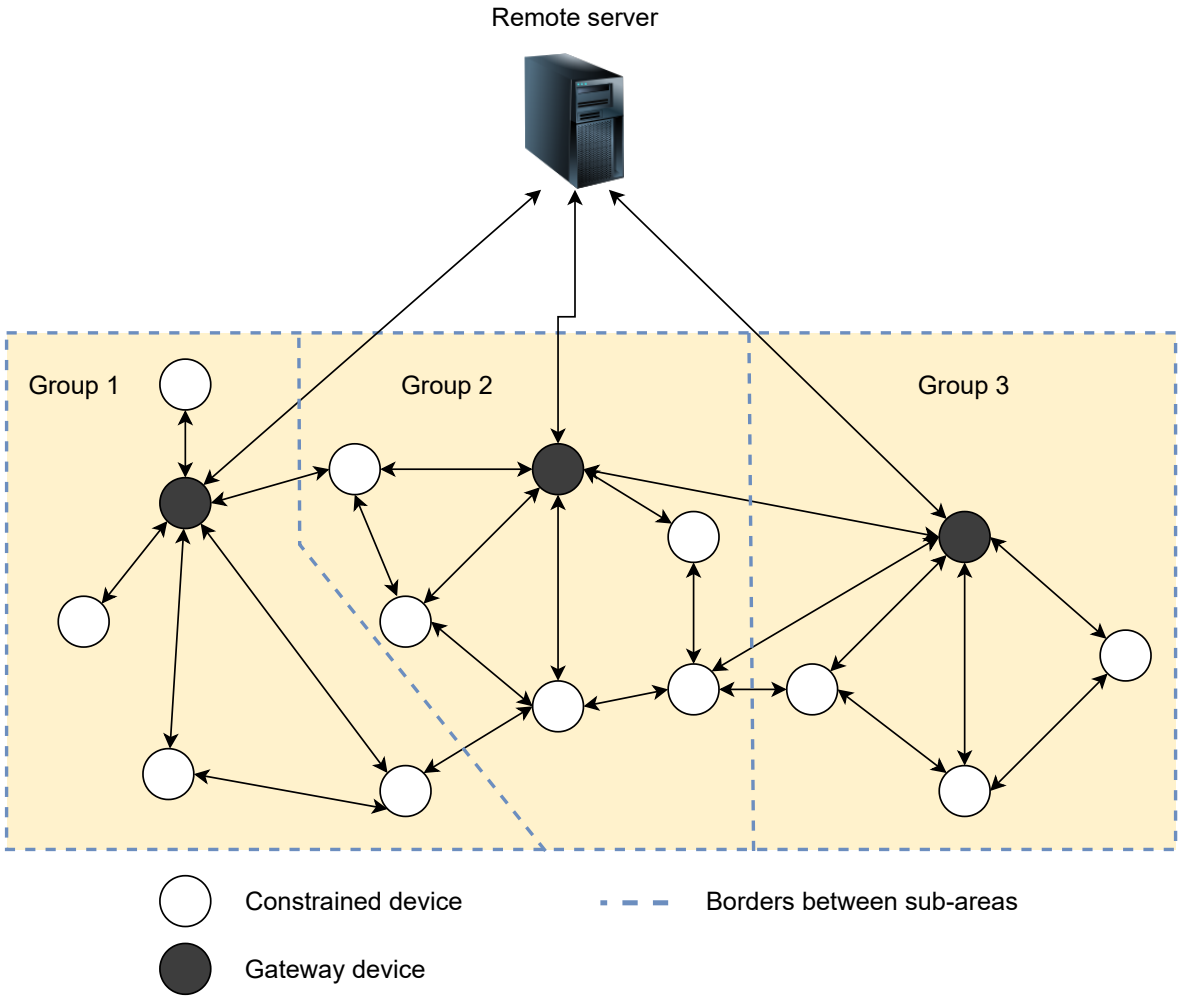


Figure 3.1: The proposed network model

3.4 Solution steps

In our approach, we outline six distinct phases: initialization, the establishment of pairwise and group-wise keys, the addition of new nodes, the revocation of keys, and key refresh. The notations used throughout this process are listed in Table 3.1.

3.4.1 Initialization phase

The secret pool encompasses randomly generated positive numbers, which are generated offline by the remote server. These numbers are then divided into m separate

Notation	Meaning
n	The overall count of nodes present within the network
d	The overall count of node neighbors
m	The count of subareas
x	The count of nodes in each subarea
N_i	The node i
a_i	The subarea i
V_p	The vector of secrets of the previous subarea.
V_c	The vector of secrets of the current subarea.
V_n	The vector of secrets of the next subarea
V_{ic}	The current vector of secrets of the subarea i
P_p	The vector of pairwise keys of the previous subarea
P_c	The vector of pairwise keys of the current subarea
P_n	The vector of pairwise keys of the next subarea
P_{ic}	The current vector of pairwise keys of the subarea i
P'	The vector of pairwise keys without nullable values
$\|P'\|$	The size of P without nullable values
I	The vector of the neighbor nodes' identifiers
$\|I\|$	The size of I
S_i	The secret of node i
K_i	The session key that has been established between the remote server and gateway i through the utilization of their public and private keys.
$K_{i,j}$	The pairwise key between node i and node j
K_g	The group key of a subarea
$V[i,j]$	The combination of j values from vector V, starting from index i (including index i). If an index included in this combination exceeds the vector's range, the combination resumes from the vector's beginning.
f_k	A hash or a lightweight symmetric encryption function using the key k.
α	A security parameter against brute force attacks
l_{gid}	The size of the group identifier
l_s	The size of the secret
l_k	The size of the key
l_{mac}	The size of the MAC
l_{type}	The size of the message type
$l_{message}$	The size of message
l_{block}	The size of encryption block
E	The size of the encrypted data
SID_k	The local identifiers of the nodes in the subset k
$SGUID_k$	The global identifiers of the nodes in the subset k
T_r	Revocation period

Table 3.1: Symbolic notations

groups, with each group holding a specific number of these random numbers, referred to as secrets. To ensure the integrity of subsequent processes, it is important that each secret is different, preventing any duplication of keys.

Furthermore, The nodes are organized into subsets, similar to how the secrets are grouped. However, there's a specific arrangement: each subset should be connected to at least one neighboring subset. For instance, the first subset connects only to the next one, the last one connects solely to the previous, and middle subsets maintain connections with both their previous and next subsets. Intriguingly, each subset must include a minimum of one gateway node, and the count of nodes within each, designated as x, equates to the number of secrets associated with that specific subset.

Now, what distinguishes each group of secrets and nodes is a unique positive number, assigned from 1 to m, where m is the total number of these groups. The server employs a random assignment procedure to allocate the secrets to the nodes within the same group based on these identifiers. Importantly, these groups might not have an equal number of nodes, meaning that the size of each group can vary.

In preparation for deployment, a crucial preliminary step is required, primarily aimed at preloading specific data into the memory of the nodes. The data to be preloaded encompass the following elements:

- **Vector Sets:** These consist of up to three distinct vectors denoted as V_p, V_c, and V_n. These vectors are populated with random and unique positive numbers. Notably, each node's Global Unique Identifier (GUID) is the result of the concatenation of its local identifier and the identifier of its associated group. Depending on the particular scenario, these vectors may contain the identifiers of nodes residing in the previous, current, and next sub-areas.
- **Group identifiers (GIDs):** Up to three group identifiers pertaining to the previous, current, and next sub-areas are included, contingent upon their existence.
- **Hash Function:** An essential component is a hash function designed to accept both plaintext and a secret as input, providing a critical cryptographic function within the system.
- **Security Parameter:** The security parameter, denoted as α, holds significance as it governs key aspects of the security protocols employed within the system.

When we refer to GUID, we imply "uniqueness across the network."

Each node is part of its respective current subset, which indicates that V_c has the key to its respective secret. The local identifiers of the nodes within a given subset are strictly non-repetitive because each node is represented as an element of its corresponding principal vector denoted as V_c. This assertion is grounded in the fundamental property of vector indexing, where the indices associated with the vectors are inherently unique. The presence of distinct indices for each vector facilitates the replication of local identifiers across multiple vectors while ensuring their non-repetition within the same vector.

The nodes are deployed randomly by dispersing them throughout the deployment area using a helicopter, as an example. The system should be partitioned into m sub-deployments, with each sub-deployment assigned to a specific subset of nodes. These sub-deployments should be executed sequentially, following the order of the nodes' identifiers. The deployment strategy arranges the subsets in a linear chain, with each subset having a previous and next subset, except for the first and last subsets, which only have one adjacent subset. From an alternative perspective, the arrangement of nodes within a subset's sub-deployment is stochastic and does not necessarily follow a linear pattern. This deployment strategy involves dividing the given area into m sub-areas, where each sub-area is assigned a specific subset of nodes. The quantity of subsets is contingent upon the characteristics of the deployment area and the available deployment capabilities. Hence, an increase in the number of options leads to a corresponding increase in the number of subsets that can be generated. Following the deployment, the sub-areas remain fixed in terms of surface area, except the first and last ones. These two sub-areas have the potential to be extended by adding additional nodes in the future. An example of the result is depicted in Figure 3.2.

Illustrative example

It is worth mentioning that the numbers presented in this particular example are intentionally kept small to enhance simplicity and ease of understanding. However, in practical application, the numerical values can be significantly larger. In this particular instance, we consider a system with a total of $n = 9$ nodes. These nodes are divided into $m = 3$ distinct subsets. The cardinality of each subset is denoted by

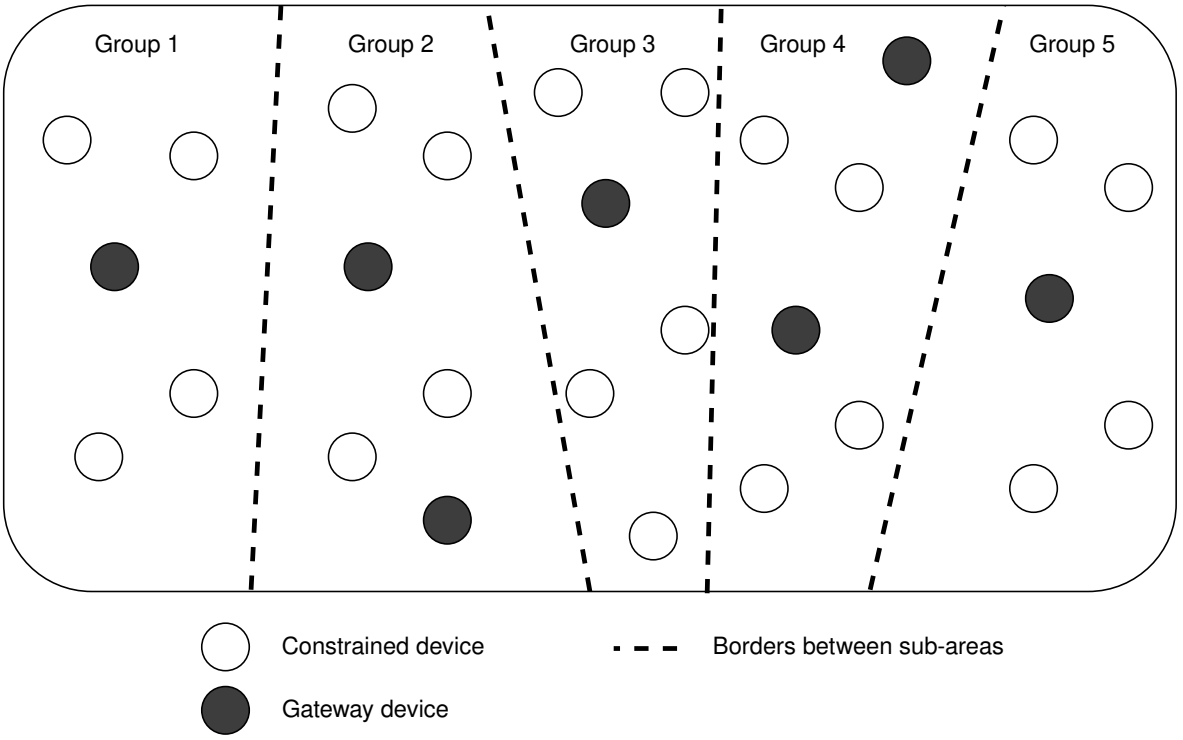


Figure 3.2: Deployment area after its division

x_k, as indicated in equation 3.1.

$$\begin{aligned} \forall k \in \mathbb{N}, 1 \leq k \leq 3 : \\ x_1 = 3 \\ x_2 = 2 \\ x_3 = 4 \end{aligned} \tag{3.1}$$

The variable k denotes the numerical identifier of the subset. The initial subset exclusively possesses a subsequent subset, the second subset possesses both a preceding and subsequent subset, whereas the third subset solely possesses a preceding subset. Each subset is assigned a distinct identifier, which is a positive number ranging from 1 to m. The subsets in the example are identified as 1, 2, and 3.

Subsets are characterized by the isolation of their local nodes' identifiers. In a given subset, it is not permissible for nodes to have duplicate local identifiers. However, it is acceptable for nodes to have duplicate local identifiers across different subsets. In this particular case, it's acceptable for nodes to possess identical local identifiers when distributed across distinct subsets. The local identifiers for these subsets symbolized as SID_k, are as follows:

$$\begin{aligned} &\forall k \in \mathbb{N}, 1 \leq k \leq 3 : \\ &\qquad SID_1 = \{1, 2, 3\} \\ &\qquad SID_2 = \{1, 2\} \\ &\qquad SID_3 = \{1, 2, 3, 4\} \end{aligned} \tag{3.2}$$

The global identifiers of nodes consist of the combination of their local identifiers and their subsets' identifiers. As shown in Figure 3.3, the global identifiers $SGUID_k$ of the subsets are provided as follows:

$$\begin{aligned} &\forall k \in \mathbb{N}, 1 \leq k \leq 3 : \\ &\qquad SGUID_1 = \{11, 12, 13\} \\ &\qquad SGUID_2 = \{21, 22\} \\ &\qquad SGUID_3 = \{31, 32, 33, 34\} \end{aligned} \tag{3.3}$$

Each node possesses a secret value, which is a randomly generated positive number provided by the server and stored in the node's memory before its deployment. The nodes' secrets within the various subsets $SSEC_k$ can be identified as follows:

$$\begin{aligned} &\forall k \in \mathbb{N}, 1 \leq k \leq 3 : \\ &\qquad SSEC_1 = \{8, 5, 7\} \\ &\qquad SSEC_2 = \{2, 6\} \\ &\qquad SSEC_3 = \{4, 1, 3, 9\} \end{aligned} \tag{3.4}$$

The three subsets can be represented by the combinations of their previous and next subsets' identifiers as follows: $(none, 2)$, $(1, 3)$, and $(2, none)$.

As demonstrated in the previous example, the vectors of the previous, current, and next subsets are as follows:

- The first subset has:
  - V_{1p} does not exist as it is the initial subset and therefore does not have any preceding subsets.
  - V_{1c} contains the nodes's secrets 8, 5, and 7, which are identified globally as 11, 12, and 13, respectively.

  - V_{1n} contains the nodes's secrets 2 and 6 which are identified globally as 21 and 22.

- The second subset has:

  - V_{2p} contains the nodes's secrets 8, 5, and 7, which are identified globally as 11, 12, and 13.
  - V_{2c} contains the nodes's secrets 2 and 6, which are identified globally as 21 and 22.
  - V_{2n} contains the nodes's secrets 4, 1, 3, and 9, which are identified globally as 31, 32, 33, and 34.

- The third subset has:

  - V_{3p} contains the nodes's secrets 2 and 6, which are identified globally as 21 and 22.
  - V_{3c} contains the nodes's secrets 4, 1, 3, and 9, which are identified globally as 31, 32, 33, and 34.
  - V_{3n} does not exist as it is the final subset and therefore does not have any subsequent subsets.

The example is illustrated in Figure 3.3.

3.4.2 Pairwise key establishment phase

This phase comprises two sub-phases: node discovery and key agreement. They are pivotal in establishing pairwise keys between nodes. In the following sub-sections, we delve into the details of each sub-phase.

Neighbors discovery

The objective of this sub-phase is to identify the neighboring nodes of each node.

Following the completion of the deployment phase, each node denoted as i initiates the discovery sub-phase by broadcasting a discovery message to all neighboring nodes

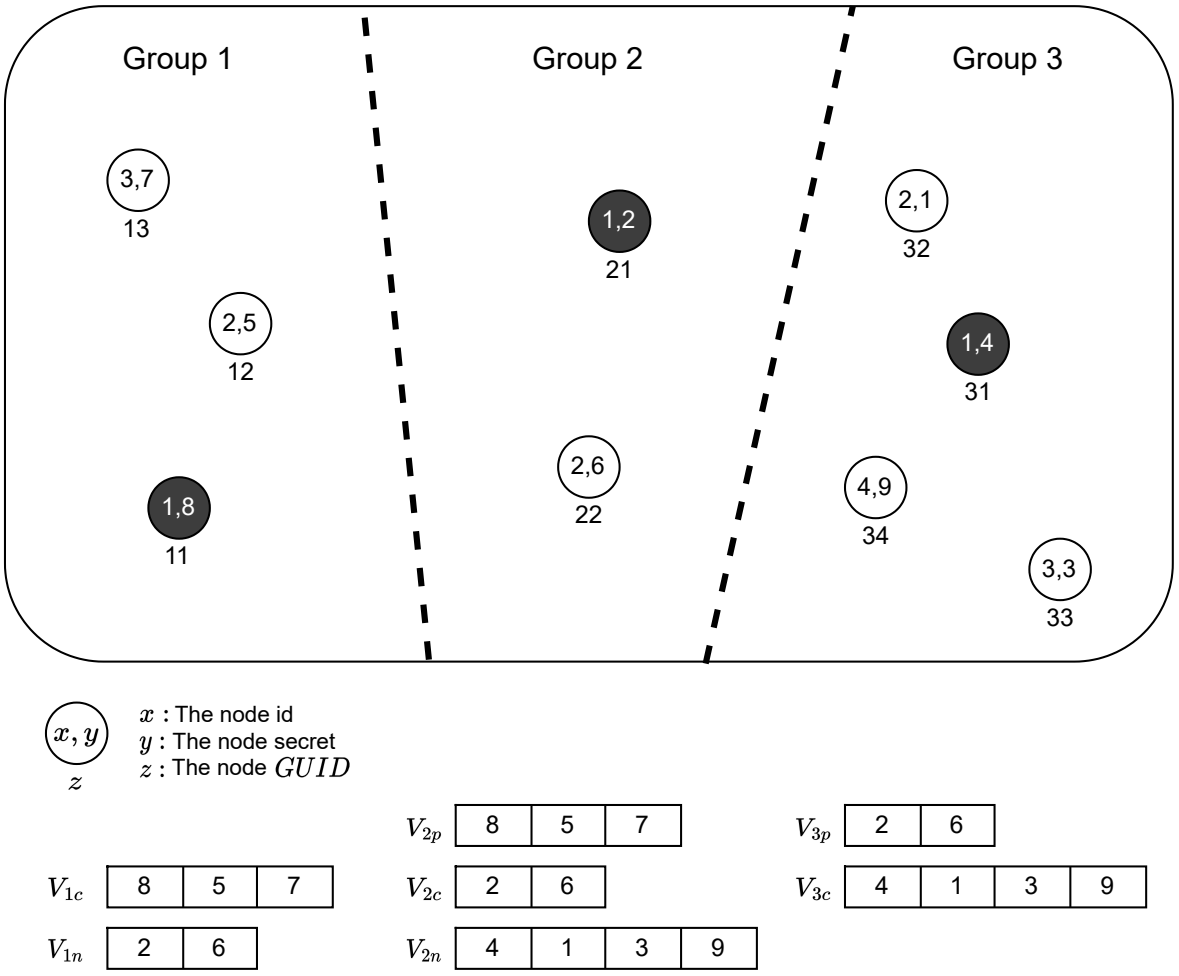


Figure 3.3: Illustrative example of the initialization phase

within its radius. Included in this message is the node's GUID, a nonce $Nonce_i$, and an encrypted digest. The encryption is done using a lightweight symmetric algorithm such as $ChaCha20$ or a hash function, depending on the desired security level. The encryption key or secret used is $V_{ic}[i, \alpha]$. The following communication is presented for consideration:

$$N_i \rightarrow N_j : Discovery, GUID_i || Nonce_i || f_{V_{ic}[i,\alpha]}(GUID_i || Nonce_i) \tag{3.5}$$

Any receiver whose physical address is lower than that of the sender will disregard the message. The remaining nodes verify the integrity of the received message by computing the digest of its first part, which consists of the concatenation of $GUID_i$

and $Nonce_i$. This computation is performed using $V_{ic}[i, \alpha]$ as a secret or an encryption key, and the resulting digest is then compared to the digest of the received message. If the comparison equality is not met, the message is disregarded. Upon successful completion of the verification process, the receiver node proceeds to include the node id in its neighbors' vector and initiates the computation of the pairwise key, as elaborated in the subsequent sub-phase. Subsequently, node j responds to node i by sending an Acknowledgment (ACK) message. This message includes the values of $GUID_j$ and $Nonce_j$, as well as a digest obtained by encrypting the concatenation of $GUID_j$ and $Nonce_j$ using the encryption key or secret $V_{jc}[j, \alpha]$. The content of the communication is as follows:

$$N_j \rightarrow N_i : ACK, GUID_j || Nonce_j || f_{V_{jc}[j,\alpha]}(GUID_j || Nonce_j) \tag{3.6}$$

The comparison of physical addresses is conducted using an alphanumeric approach. This operation prevents nodes from initiating discovery with other nodes that have already initiated it, due to the asynchronous nature of distributed systems.

Upon receiving a response from node j, node i proceeds to verify the message's integrity. This is done by calculating the digest of the first part of the message, which consists of the concatenation of $GUID_j$ and $Nonce_j$. The encryption key or secret used for this calculation is denoted as $V_{jc}[j, \alpha]$. The calculated digest is then compared with the digest received in the message to ensure their consistency. If the outcome of the equality comparison is unsuccessful, the message is disregarded. In contrast, if the transmission is successful, the receiving node proceeds to compute the pairwise key K_{ij}, as described in the subsequent sub-phase 3.4.2, and then includes node j in the vector of pairwise neighbors.

Key agreement

Within the context of the key agreement sub-phase, every node engages in the computation of a pairwise key. This computation involves the multiplication of the confidential values held by the two respective nodes, followed by the encryption of the resultant product through a preconfigured lightweight encryption function. The outcome of this multiplication operation serves as the shared secret, from which the pairwise key is subsequently derived. The pairwise key calculation process is as fol-

lows:

$$K_{ij} = f_{V_{ic}[i,\alpha]*V_{jc}[j,\alpha]}(V_{ic}[i,\alpha] * V_{jc}[j,\alpha]) \tag{3.7}$$

For every establishment of a pairwise key between two nodes denoted as i and j, the resulting pairwise key is allocated storage within one of the neighbor vectors, namely P_p, P_c, or P_n. This allocation is contingent upon the sub-area affiliation of the other node involved in the key establishment process. To elucidate this, suppose node i is situated within the first sub-area while node j belongs to the second sub-area. In this scenario, node i archives the pairwise key in vector P_n, while node j retains it within vector P_p.

It is noteworthy that each node i preserves its private secret S_i within vector P_c, where the index corresponds to its local identifier, denoted as i.

Furthermore, it is important to observe that if node j, hailing from the prior, current, or subsequent sub-area concerning node i, is not classified as a neighbor of i, then i undertakes the task of populating the associated index within the respective neighbor vector P in the subsequent manner:

- If there exists at least one neighboring node denoted as k of node i within the designated sub-area of node j, where the local identifier Id_k is greater than that of j, node i proceeds to populate the corresponding index within the respective neighbor vector P with a null value.
- Conversely, if no neighbor node k of node i within the assigned sub-area of node j possesses a local identifier Id_k greater than that of j, node i takes the action of resizing the corresponding neighbor vector P to accommodate a maximum index value of $Id_j - 1$.

If the size of any of the P vectors containing nullable values exceeds that of the corresponding P vector devoid of nullable values, in addition to the size of its corresponding index values, node i takes the following actions:

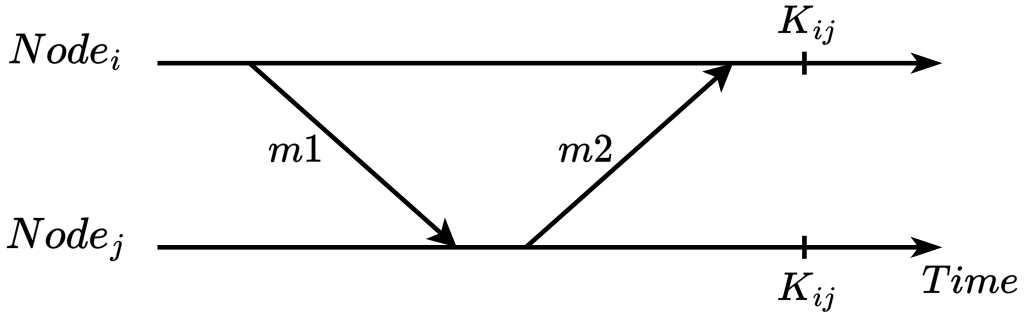
Node i stores a P vector without the nullable values, and concurrently, it retains another vector containing the corresponding index values, as represented in equation

(3.8). This operation serves to potentially reduce storage requirements, particularly in scenarios where a node has limited coverage within its primary and neighboring sub-areas.

$$\begin{cases} P', I & \text{if } |P| > |P'| + |I| \\ P & \text{else} \end{cases} \tag{3.8}$$

In Figure 3.4, we depict the interactions between two nodes denoted as i and j during the pairwise key establishment phase. To initiate the neighbors' discovery sub-phase, node i broadcasts a discovery message referred to as $m1$ to all nodes within its communication range. The message $m1$ includes critical information such as the node identifier $GUID_i$, a nonce denoted as $Nonce_i$, and its corresponding Message Authentication Code (MAC). Upon reception of this message, node j is responsible for verifying its integrity by computing the message digest and subsequently comparing it with the received MAC value.

Following the successful integrity verification process, node j proceeds to send an ACK message designated as $m2$, which incorporates its node identifier $GUID_j$, the nonce $Nonce_j$, and its corresponding Message Authentication Code MAC. Subsequently, both nodes, i and j, enter the key agreement sub-phase, where they perform the necessary calculations to derive their pairwise key denoted as K_{ij}.



$$m1 : Discovery, GUID_i||Nonce_i, MAC(GUID_i||Nonce_i, V_{ic}[i])$$

$$m2 : ACK, GUID_j||Nonce_j, MAC(GUID_j||Nonce_j, V_{jc}[j])$$

Figure 3.4: Pairwise key establishment phase

Figure 3.5 illustrates the pairwise key establishment process between two nodes i and its neighbor j.

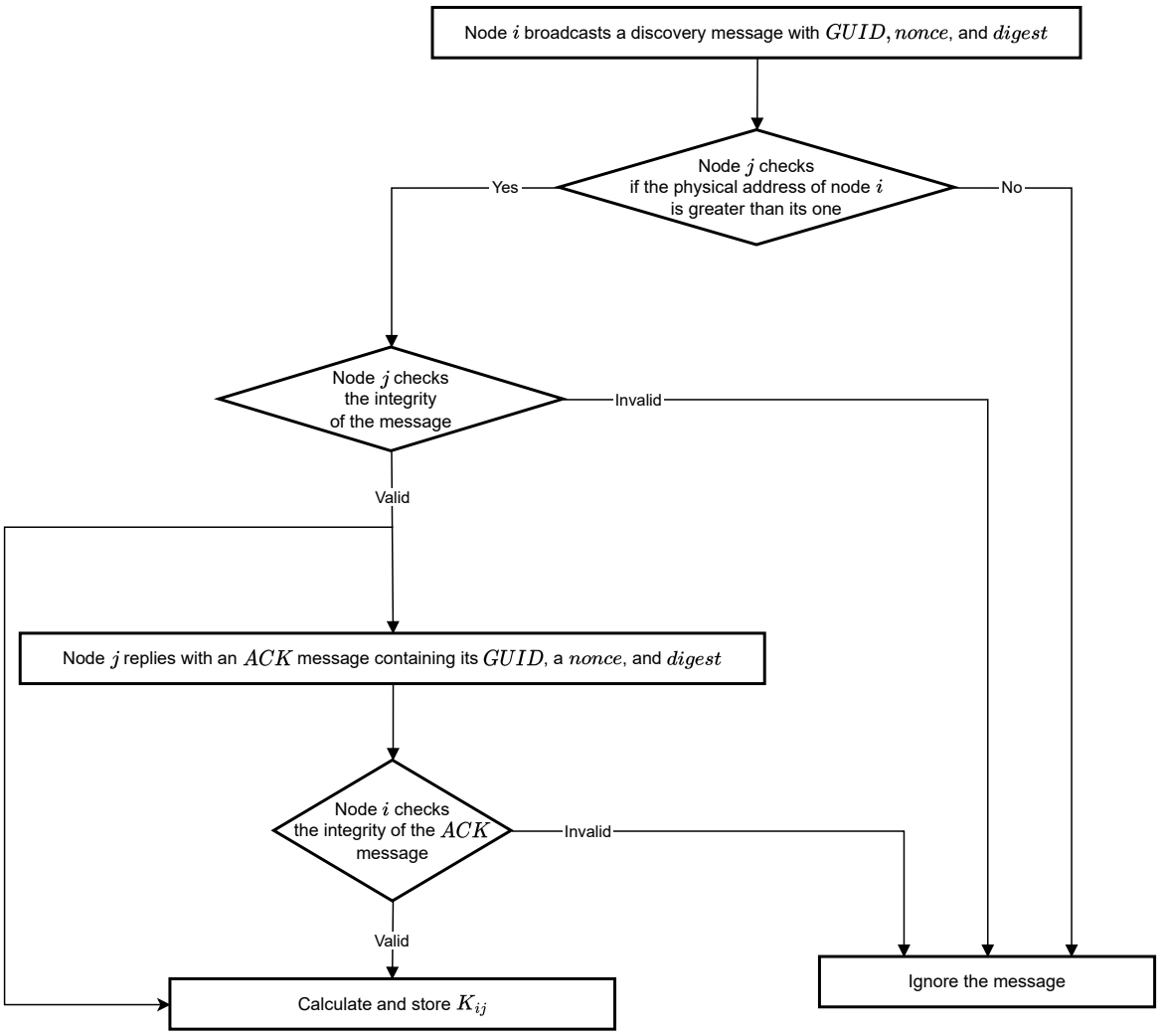


Figure 3.5: Process of establishing a pairwise key between two nodes

Illustrative example

This example is built upon the preceding initialization phase's 3.4.1. For the sake of simplicity, we consider a security parameter α set to 1. Let's consider an example involving the establishment of pairwise keys for specific nodes, namely nodes 11, 12, 22, and 34. Each of these nodes transmits a message containing the following:

$$N_{11} \rightarrow * : Discovery, 11||15||f_8(11||15)$$

$$N_{12} \rightarrow * : Discovery, 12||19||f_5(12||19)$$

$$N_{22} \rightarrow * : Discovery, 22||9||f_6(22||9)$$

$$N_{34} \rightarrow * : Discovery, 34||14||f_9(34||14)$$

Upon receiving and verifying the broadcasted messages, we consider the neighbor nodes of each node as follows:

$$N_{11} : \{N_{12}, N_{13}, N_{21}, N_{22}\}$$

$$N_{12} : \{N_{11}, N_{13}, N_{21}, N_{22}\}$$

$$N_{22} : \{N_{11}, N_{12}, N_{13}, N_{21}\}$$

$$N_{34} : \{N_{21}, N_{22}, N_{31}, N_{32}, N_{33}, N_{34}\}$$

Each of these nodes then proceeds to calculate its pairwise keys as follows:

$$K_{11,12} = K_{12,11} = f_{8*5}(8 * 5) = f_{5*8}(5 * 8) = f_{40}(40)$$

$$K_{11,13} = K_{13,11} = f_{8*7}(8 * 7) = f_{7*8}(7 * 8) = f_{56}(56)$$

$$K_{11,21} = K_{21,11} = f_{8*2}(8 * 2) = f_{2*8}(2 * 8) = f_{16}(16)$$

$$K_{11,22} = K_{22,11} = f_{8*6}(8 * 6) = f_{6*8}(6 * 8) = f_{48}(48)$$

$$K_{12,13} = K_{13,12} = f_{5*7}(5 * 7) = f_{7*5}(7 * 5) = f_{35}(35)$$

$$K_{12,21} = K_{21,12} = f_{5*2}(5 * 2) = f_{2*5}(2 * 5) = f_{10}(10)$$

$$K_{12,22} = K_{22,12} = f_{5*6}(5 * 6) = f_{6*5}(6 * 5) = f_{30}(30)$$

$$K_{22,13} = K_{13,22} = f_{6*7}(6 * 7) = f_{7*6}(7 * 6) = f_{42}(42)$$

$$K_{22,21} = K_{21,22} = f_{6*2}(6 * 2) = f_{2*6}(2 * 6) = f_{12}(12)$$

$$K_{22,31} = K_{31,22} = f_{6*4}(6 * 4) = f_{4*6}(4 * 6) = f_{24}(24)$$

$$K_{22,32} = K_{32,22} = f_{6*1}(6 * 1) = f_{1*6}(1 * 6) = f_{6}(6)$$

$$K_{22,33} = K_{33,22} = f_{6*3}(6 * 3) = f_{3*6}(3 * 6) = f_{18}(18)$$

$$K_{22,34} = K_{34,22} = f_{6*9}(6 * 9) = f_{9*6}(9 * 6) = f_{54}(54)$$

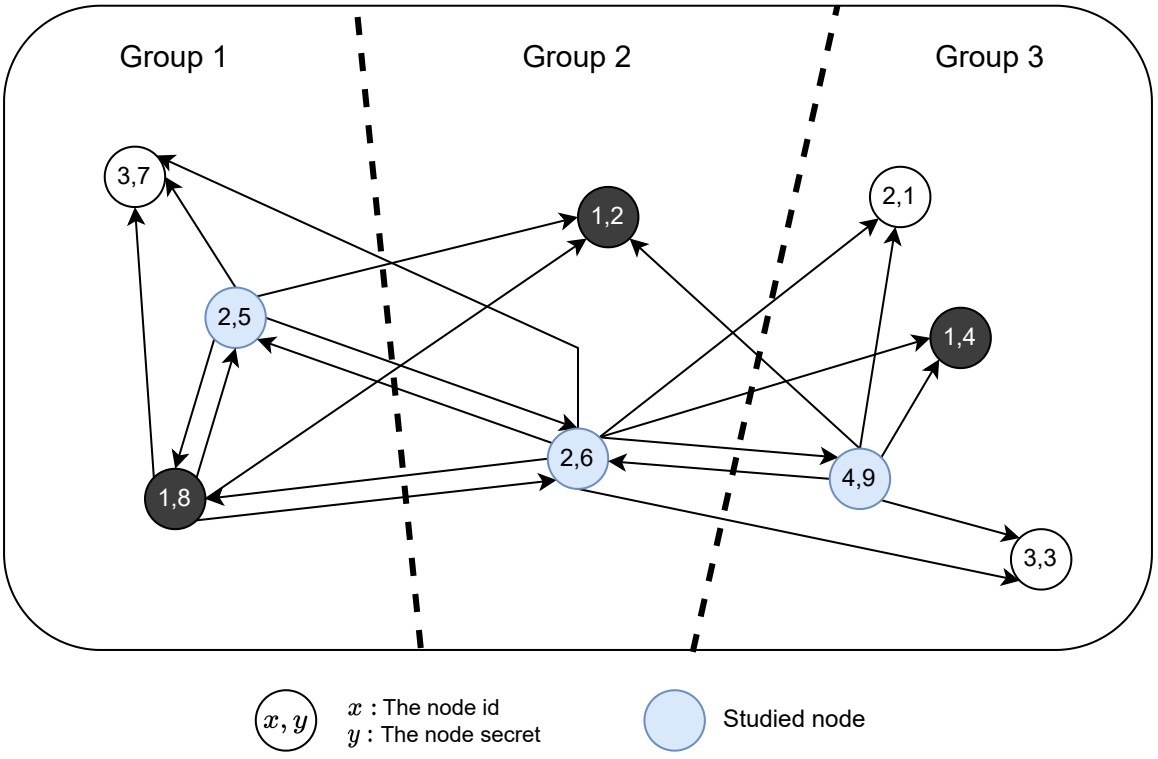
$$K_{34,21} = K_{21,34} = f_{9*2}(9 * 2) = f_{2*9}(2 * 9) = f_{18}(18)$$

$$K_{34,31} = K_{31,34} = f_{9*4}(9 * 4) = f_{4*9}(4 * 9) = f_{36}(36)$$

$$K_{34,32} = K_{32,34} = f_{9*1}(9 * 1) = f_{1*9}(1 * 9) = f_9(9)$$
$$K_{34,33} = K_{33,34} = f_{9*3}(9 * 3) = f_{3*9}(3 * 9) = f_{27}(27)$$

The computed pairwise keys are stored in the nodes' memory and utilized in the subsequent phases of the protocol.

Figure 3.6 depicts the pairwise key establishment phase.



N_{11}: P_c | 8 | $f_{40}(40)$ | $f_{56}(56)$ | P_n | $f_{16}(16)$ | $f_{48}(48)$ |

N_{12}: P_c | $f_{40}(40)$ | 5 | $f_{35}(35)$ | P_n | $f_{10}(10)$ | $f_{30}(30)$ |

N_{22}: P_p | $f_{48}(48)$ | $f_{30}(30)$ | $f_{42}(42)$ | P_c | $f_{12}(12)$ | 6 | P_n | $f_{24}(24)$ | $f_6(6)$ | $f_{18}(18)$ | $f_{54}(54)$ |

N_{34}: P_p | $f_{18}(18)$ | $f_{54}(54)$ | P_c | $f_{36}(36)$ | $f_9(9)$ | $f_{27}(27)$ | 9 |

Figure 3.6: Example of establishing a pairwise key

3.4.3 Groupwise key establishment phase

The increase in communication cost has a notable impact on performance and energy metrics, particularly when transmitting identical data to multiple nodes. To mitigate this, we have employed a method for identifying node neighbors by combining the use of the MAC with certain plaintext data. However, it is important to note that the practicality of this technique is limited due to the unclear distinction between public and private data.

To address this challenge, the adoption of a group-wise key becomes imperative. With such a key in place, a group of nodes can collaborate efficiently to avoid redundant message content transmission. In this context, each node, denoted as i, belonging to the same group as the gateway g_i, computes an identical group-wise key according to the following equation:

$$Kgi = f_{V_{ic}[g]}(V_{ic}[0]||V_{ic}[V_{ic}.length - 1]||V_{ic}[g]) \tag{3.9}$$

Before encryption, it combines both its current secret vector's first and last elements with the gateway secret, utilizing the latter as the encryption key.

Illustrative example

Expanding upon the preceding illustration example, it's important to note that all the nodes, including the gateways, preserve identical secrets within their memory.

For each gateway node G_k and all the nodes within its sub-area that include it in their list of neighbors, the calculation of the group-wise key proceeds as follows:

$$\begin{aligned} \forall k \in \mathbb{N}, 1 \leq k \leq 3 : \\ K_{G_1} &= f_8(8||7||8) = f_8(878) \\ K_{G_2} &= f_2(2||6||2) = f_2(262) \\ K_{G_3} &= f_4(4||9||4) = f_4(494) \end{aligned} \tag{3.10}$$

Once the group-wise keys are computed and stored, the constrained nodes proceed to clear their memory of the secret vectors. However, they retain the vectors related to pairwise keys, denoted as P_p, P_c, and P_n, they retain the identifiers of various

neighboring groups as well. These retained data will prove useful in subsequent phases of the operation.

In contrast, the gateway nodes preserve all of this information in their memory, as it is essential for use in the upcoming phases. This decision stems from the fact that gateway nodes have secured their memory through the use of TPM, which is not available to the constrained nodes.

3.4.4 New node addition phase

In the context of the new node addition phase, the server takes the initiative by identifying the accessible gateway node within the newly deployed sub-area of the node network. Subsequently, it generates a fresh pairwise key denoted as K_{ng_i}, intended to secure the communications between the gateway node and the new node. Alongside this, the server also generates a random positive numeric secret.

The gateway node plays a pivotal role in this phase by incorporating the newly generated secret into its existing secret vector. Subsequently, it responds to the server's action by transmitting an ACK message. This message includes the vectors V_p, V_c, and V_n, which collectively represent the secrets of the sub-area.

In parallel, the server assigns a unique local identifier to the new node. This identifier is calculated as $count(V_{ic}) + 1$, signifying the count of nodes within its respective sub-area, which in turn becomes the identification number for the new node. Additionally, the server associates this new node with the group identifier of the gateway.

Moreover, the server preloads the generated keys onto the new node and proceeds to dispatch an encrypted message. This message contains K_{ng_i} and the secret designated for the new node, further facilitating secure communication between the new node and the gateway node.

The interactions between the server and the gateway node are formalized through equations (3.11) and (3.12), encapsulating the following exchanges:

$$S \rightarrow G_i : \{REQ, Nonce_n, S_n, K_{ng_i}\}_{K_i} \tag{3.11}$$

$$G_i \rightarrow S : \{ACK, Nonce_n, V_p, V_c, V_n\}_{K_i} \tag{3.12}$$

Upon successful deployment of the new node, it initiates communication by transmitting an $INFORM$ message to the reachable gateway node within its designated sub-area. This message serves as a notification of the successful deployment of the new node to the gateway.

$$N_n \rightarrow G_i : \{INFROM, Nonce_n\}_{K_{ng_i}} \tag{3.13}$$

The gateway undergoes a calculation process to derive the new group-wise key. This calculation involves hashing its previous group key, utilizing the newly acquired secret associated with the new node.

$$K_g = Hash(K_g, S_n) \tag{3.14}$$

In order to safeguard the network from potential forward privacy attacks, a hash function is employed in the calculation of the new group key, as depicted in equation (3.14). This security provision guarantees that, should an attacker successfully compromise a node and acquire its current group key, the attacker would still lack the means to reconstruct the old group key. The old group key is essential for decrypting previous messages, rendering it inaccessible to malicious actors.

Subsequently, the gateway proceeds to disseminate the secret associated with the new node to nodes residing within its current, previous, and next sub-areas, contingent upon their presence. Importantly, this message is encrypted using the old group-wise key as a security measure.

$$G_i \rightarrow * : \{INFROM, Nonce_g, S_n\}_{K_g} \tag{3.15}$$

Each of the nodes within this group employs the equation (3.14) to compute the new group-wise key. Subsequently, they transmit an ACK message back to the gateway as a confirmation of the successful reception of the previous message. This step ensures that the newly calculated group-wise key is synchronized among all relevant nodes.

$$N_j \rightarrow G_i : \{ACK, Nonce_j\}_{K_{jg_i}} \tag{3.16}$$

The gateway proceeds by transmitting an ACK message to the new node, signifying that the discovery process is prepared to be started. This message also encapsulates the newly established group-wise key, ensuring that the new node is equipped with the necessary information to proceed with the discovery process.

$$G_i \rightarrow N_n : \{ACK, Succ(Nonce_g), K_g\}_{K_{ng_i}} \tag{3.17}$$

Subsequently, the new node initiates the discovery process by dispatching a discovery message to all of its neighboring nodes. This communication employs the same pairwise key establishment technique as previously employed.

$$\begin{aligned} N_n \rightarrow * : & Discovery, \\ & GUID_i || Succ(Nonce_n), \\ & f_{S_n}(GUID_n || Succ(Nonce_n)) \end{aligned} \tag{3.18}$$

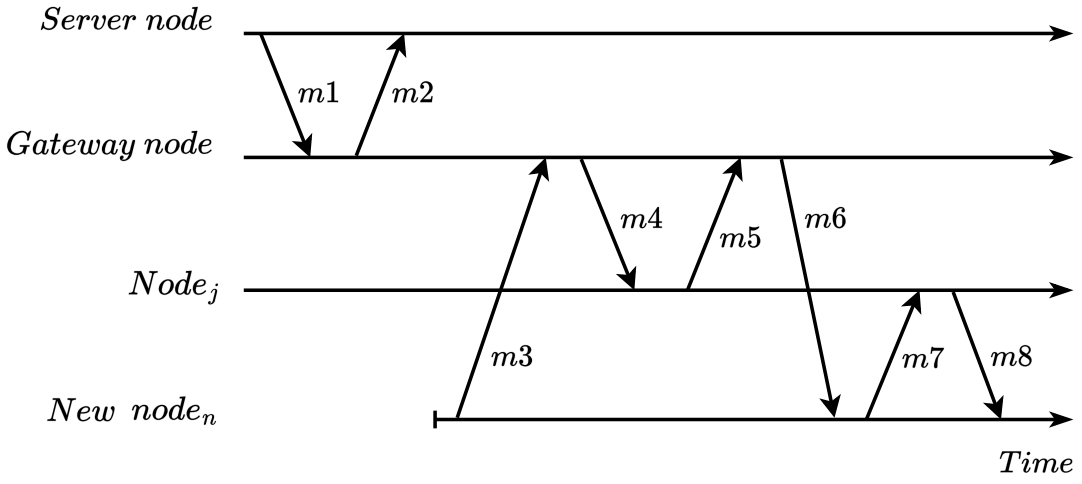
The neighboring nodes that receive the discovery message from the new node undertake the calculation of pairwise keys for each relevant connection (new node, neighboring node). Following this calculation, they respond by transmitting ACK messages back to the new node.

$$\begin{aligned} N_j \rightarrow N_n : & ACK, \\ & GUID_j || Succ(Nonce_j), \\ & f_{S_j}(GUID_j || Succ(Nonce_j)) \end{aligned} \tag{3.19}$$

The new node proceeds by computing a pairwise key for each pair, consisting of the new node and its respective neighbor node.

$$K_{nj} = f_{S_n * S_j}(S_n * S_j) \tag{3.20}$$

A graphical illustration of the new node addition phase is presented in Figure 3.7, offering a clear depiction of the key steps involved in this process.



$m1 : \{REQ, Nonce_n, S_n, K_{ng_i}\}_{K_i}$

$m2 : \{ACK, Nonce_n, V_p, V_c, V_n\}_{K_i}$

$m3 : \{INFROM, Nonce_n\}_{K_{ng_i}}$

$m4 : \{INFROM, Nonce_g, S_n\}_{K_g}$

$m5 : \{ACK, Nonce_j\}_{K_{jg_i}}$

$m6 : \{ACK, Succ(Nonce_g), K_g\}_{K_{ng_i}}$

$m7 : ACK, GUID_n||Succ(Nonce_n), f_{S_n}(GUID_n||Succ(Nonce_n))$

$m8 : ACK, GUID_j||Succ(Nonce_j), f_{S_j}(GUID_j||Succ(Nonce_j))$

Figure 3.7: New node addition phase

Illustrative example

In the following illustrative example, which is the continuation of the previous examples, we explore the process of adding a new node to the network. To demonstrate this, let's assume that we are deploying a new node within sub-area 3 of the network. Here are the detailed steps:

- **Selection of Reachable Gateway:** The server chooses the reachable gateway node with $GUID = 31$ as the new node's gateway. It generates a new pairwise key $K_{ng_{31,35}}$ for secure communication and assigns a random positive number secret $S_n = 10$.
- **Preloading Keys:** Following communication with the gateway node, the server preloads the generated keys onto the new node and sends them encrypted in message $m1$ to the same gateway.
- **Deployment Notification:** Once the new node is successfully deployed, it sends an $INFORM$ message $m3$ to its nearest gateway to notify it of its deployment.
- **Calculation of New Group-wise Key:** The gateway calculates the new group-wise key K_{g_3} by hashing its old group key using the new node's secret: $K_{g_3} = Hash(494, 10)$.
- **Broadcasting the New Node's Secret:** The gateway broadcasts message $m4$ to nodes in its current, previous, and next sub-areas. In this case, it sends the message to N_{21}, N_{22} for the previous sub-area and N_{32}, N_{33}, N_{34} for the current sub-area.
- **Confirmation and Key Calculation:** Each node receiving the broadcast message calculates the new group-wise key using the same equation as specified in Equation 3.14. They respond with an ACK message $m5$ to confirm the reception of the broadcasted message.
- **Launch of Discovery Process:** The gateway sends an ACK message $m6$ to the new node to indicate that the discovery process is ready to begin. This message also contains the new group-wise key $K_{g_3} = Hash(494, 10)$.
- **Discovery Message Broadcast:** Utilizing the pairwise key establishment technique, the new node broadcasts a discovery message $m7$ to all its neighboring nodes.

- **Neighbor Responses:** Neighbor nodes that receive the discovery message calculate pairwise keys for each pair (new node, neighbor node) and respond with ACK messages $m8$.
- **Pairwise Key Calculation:** The new node calculates pairwise keys for each of these pairs.

This process is visually represented in Figure 3.8 to provide a clear understanding of the new node addition phase in action.

3.4.5 Key refresh phase

In the "Key refresh phase," nodes undertake the crucial task of refreshing their secret materials including secrets, pairwise or group-wise keys to enhance network security and guard against potential attacks. This key refresh process is initiated under four distinct circumstances:

- **New Node Addition:** When a new node joins the network, all nodes must refresh their group-wise keys, as elaborated in the new node addition phase.
- **Node Leaving:** In instances where an existing node departs from the group or the network, the responsible gateway takes action by generating and dispatching $d-1$ messages to its non-departed neighbors (a message to each of them). These messages contain a random positive number denoted as r. Each receiving node calculates a new group-wise key using the function $K_g = f_r(K_g)$ and updates its pairwise keys using $K_{ij} = f_r(K_{ij})$. Additionally, it modifies its secret following: $S_i = S_i * r$.
- **Periodic Refresh:** At regular intervals of time, every node autonomously refreshes its pairwise and group-wise keys by applying a hash function to the existing keys. This process is conducted to ensure that the keys remain resilient and robust against potential attacks, particularly those that rely on the exploitation of the keys using brute force.
- **Refresh Request:** In response of each node to a refresh request issued by its gateway, each node is obligated to refresh its pairwise or group-wise keys.

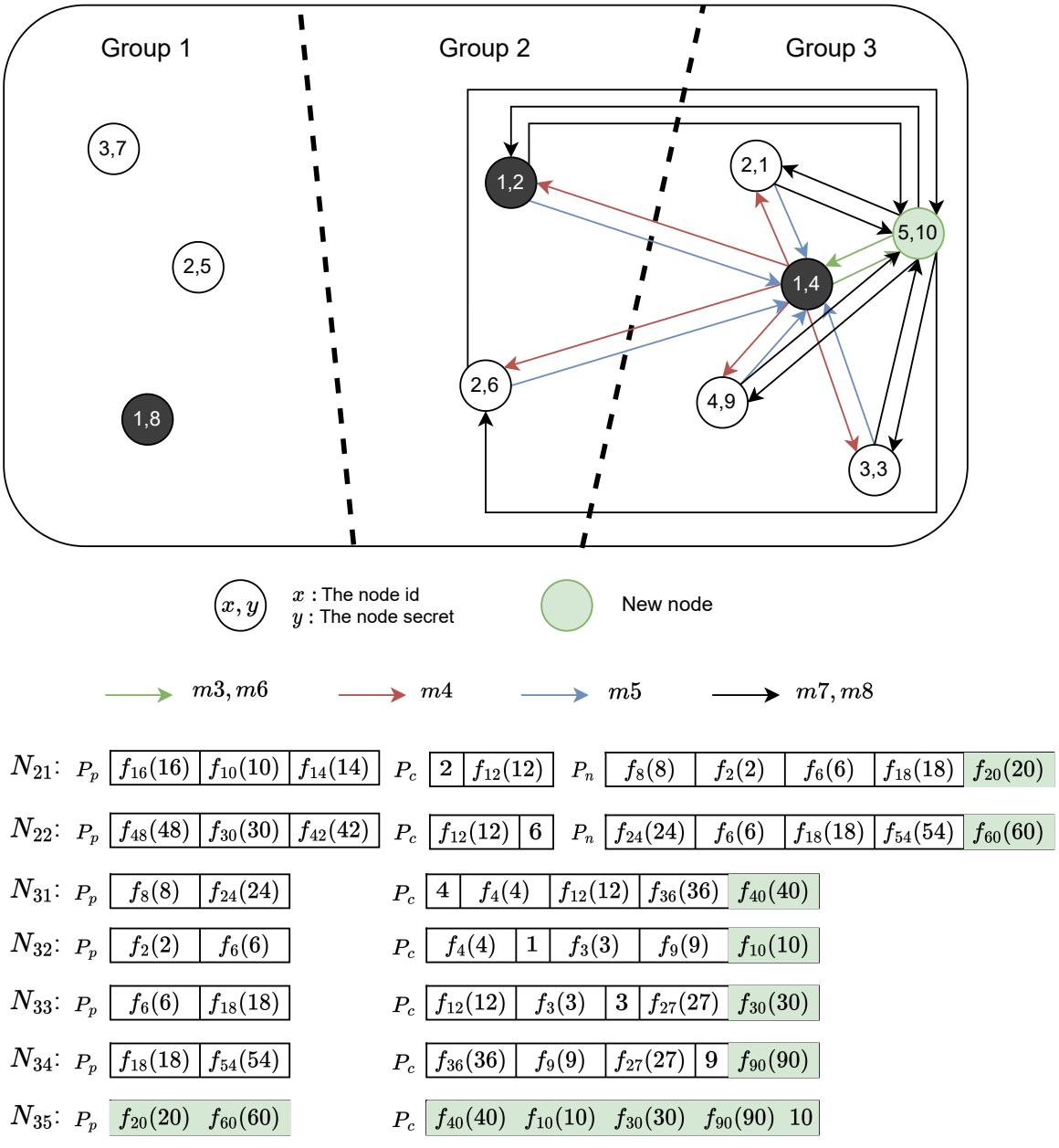


Figure 3.8: New node addition phase example

These key refresh procedures are instrumental in maintaining the security and integrity of the IoT network, ensuring that cryptographic keys remain robust and resilient against potential threats.

3.4.6 Key revocation phase

In the "Key revocation phase," the network accommodates its dynamic nature by allowing any node to exit the network, particularly when it operates with a limited battery capacity. When a node intends to leave the network under such circumstances, it initiates a leave request to its neighboring or interconnected nodes. The objective is to optimize memory usage by removing the old and deprecated keys from nodes that rely on them. Simultaneously, it transmits a group-level $LEAVE$ message to notify its departure.

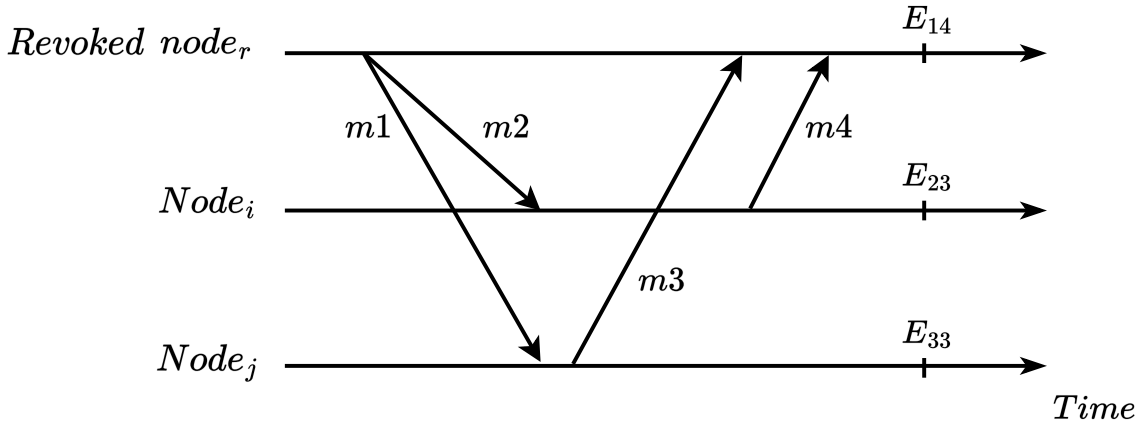
Each recipient node, upon receiving the leave request, responds with an ACK message, signifying its compliance with the key erasure task and the necessary updates to its neighbor vectors. Once the majority of nodes have responded, the departing node proceeds to clear its memory, which includes keys and other pertinent information. This process is visually depicted in Figure 3.9.

Another scenario for key revocation occurs when a node remains inactive or unreachable for a duration equal to or longer than T_r. In this case, its neighboring nodes take the initiative to delete its information from their memory and update their neighbor vectors accordingly. This periodic operation serves to enhance memory efficiency within the network.

3.5 Evaluation

3.5.1 Security analysis

In this section, we undertake a security analysis of our scheme which considers both formal and informal. It commences with an informal analysis, followed by a formal assessment. Additionally, a comparative examination of related works from a security standpoint is provided. The notation used in the preceding section is consistently applied throughout this analysis.



$m1, m2 : \{Leave, GUID_r, Nonce_r\}_{K_{gr}}$

$m3 : \{Ack, Nonce_j\}_{K_{jr}}$

$m4 : \{Ack, Nonce_i\}_{K_{ir}}$

$E_{14} : Erase\ memory$

$E_{23}, E_{33} :\ Delete\ node_r\ pairwise\ key$

Figure 3.9: Node revocation phase

Informal security analysis

This section is dedicated to conducting an informal analysis of the security aspects of our scheme. Our analysis begins by outlining the security goals, followed by an examination of the security properties.

Security goals Our scheme is built upon a set of core security goals, each addressing critical aspects of IoT network security:

- **Confidentiality:** This goal revolves around safeguarding sensitive information from unauthorized access during transmission. It is important to consider that adversaries cannot decipher the messages transmitted by legitimate nodes. Breaching confidentiality can result in severe consequences, spanning a wide range of security breaches, including data theft. The risk escalates when sensitive data, such as cryptographic keys, is involved, as compromising them could lead to violations of other security goals and properties. An example of a confidentiality attack is eavesdropping.
- **Integrity:** Integrity is a paramount concern that revolves around preventing unauthorized alterations to data. It entails ensuring that adversaries cannot tamper with the messages originating from legitimate nodes. Violating integrity can have catastrophic repercussions on the system, as it opens the door to malicious data manipulation. Man-In-The-Middle attacks exemplify attempts to compromise integrity.
- **Authentication:** Authentication is the process of verifying the identity of communicating nodes, guaranteeing that the node involved in communication is indeed the entity it claims to be. It is essential to thwart adversaries' attempts to impersonate legitimate nodes. Replay attacks represent a prevalent form of authentication attack.
- **Forward Secrecy:** Forward secrecy serves to ensure that past communications remain secure, even if an adversary manages to obtain a new secret key. In essence, it dictates that adversaries should be incapable of decrypting messages sent by legitimate nodes in the past.
- **Backward Secrecy:** Similar to forward secrecy but with a nuanced difference, backward secrecy aims to safeguard current and future communications

against adversaries who might gain access to an old secret key. Consequently, adversaries must not possess the capability to decrypt messages transmitted by legitimate nodes in the future. This safeguard is crucial for preserving the long-term security of the network.

Security properties Expanding upon the security properties of our scheme, we delve into the specifics of how it addresses critical security aspects:

- **Confidentiality:** In the first scenario, during the pairwise key establishment phase, our scheme addresses confidentiality concerns effectively.
  While adversaries can eavesdrop on messages exchanged by legitimate nodes, they cannot access sensitive information within these messages. The reason behind this is that our scheme refrains from exchanging secrets between nodes, as these secrets are preloaded into the nodes' memories. Consequently, the nodes only share public information, including group and node identifiers, along with a MAC for message integrity. Adversaries are thwarted from reading sensitive data since they lack knowledge of the MAC encryption secret used by legitimate nodes.
  In the other scenarios, the exchanged messages between different node types are fully encrypted, rendering them inaccessible to adversaries.

- **Integrity:** The scheme ensures message integrity even in the face of eavesdropping adversaries. The structure of our messages fortifies integrity in two ways.

  - First, the second part of each message is encrypted using the first part, and the MAC authenticates the first part. Attempting to modify the first part of a message by an adversary would invalidate the MAC.
  - Second, the secret key encrypts the second part of the message, making it inaccessible to non-legitimate nodes. This same principle applies when pairwise or group-wise keys are used, as the encryption provided by these keys safeguards message integrity. Non-legitimate nodes are devoid of knowledge of these keys, rendering them ineffective in tampering with the messages.

- **Authentication:** Authentication is crucial, and our scheme provides robust protection against impersonation attempts in different scenarios:

  - **Case 1:** Before establishing a pairwise key between two nodes, adversaries cannot successfully impersonate legitimate nodes. The nodes mutually authenticate each other using their public local and group identifiers, alongside a MAC of the message containing this information. The encryption secret behind the MAC remains unknown to non-legitimate nodes, preventing them from generating a valid MAC for impersonation.
  - **Case 2:** Upon the successful establishment of a pairwise key between two nodes, the risk of impersonation is eliminated. The nodes authenticate each other using the same unique key, and the messages they exchange are encrypted exclusively for these two nodes, leaving no room for impersonation.
  - **Case 3:** In scenarios involving group-wise keys, the nodes within the group authenticate each other using this key. Messages exchanged among them are encrypted by the same group-wise key, a secret inaccessible to any nodes outside the group, thus thwarting impersonation attempts.

- **Replay Attacks:** Our scheme effectively mitigates replay attacks. Even if adversaries gain access to messages sent by legitimate nodes, replaying these messages is futile. Each message incorporates a unique nonce encrypted using $MACs$ as well as pairwise and group-wise keys before and after their key establishment processes. These encryption measures safeguard the nonce, rendering any attempts by adversaries to read or alter it ineffective.

- **Forward Secrecy:** Forward secrecy is a safeguard in our scheme, ensuring that new nodes cannot decrypt old messages exchanged by group members before their inclusion. The establishment of a new group-wise key upon adding new nodes results in a key distinct from the old one. Consequently, the new nodes remain unaware of the old group-wise key, preventing them from decrypting messages sent before their inclusion in the group.

- **Backward Secrecy:** Similar to forward secrecy, backward secrecy is upheld by our scheme. When old nodes leave or are compromised, they cannot decrypt messages sent by new group members after their departure. The independent establishment of a new group-wise key among new group members results in a different key from the old one. Additionally, the old group-wise key remains unknown to the departing nodes, precluding them from decrypting messages sent after their exit from the group. This safeguard ensures the continued security of the network even in the face of node departures or compromises.

Formal security analysis

Within this section, BAN logic [19], [26] is employed to formally evaluate the security of our protocol. This assessment provides a comprehensive analysis of the formal protocol's security and its alignment with our predefined security goals. To carry out this assessment, we employ the following notation:

- S and R represent principals within the protocol.
- T and U denote statements that are pertinent to the security analysis.

Notation Table 3.2 provides an overview of the notations utilized in this section.

Notation	**Meaning**
$S \mid\equiv R$	Principal S believes what is communicated by R.
$S \triangleleft R$	Principal S can see the messages of R.
$S \mid\sim R$	Principal S explicitly attributed a message to R.
$S \Rightarrow R$	Principal S has authority or control over R.
$\#(T)$	The variable T is unique, in other words, T is considered fresh or newly generated.
(T, U)	Concatenation of statements T and U.
$\{T\}_U$	Variable T is encrypted using key U.
$S \overset{k}{\leftrightarrow} R$	Principals S and R share the secret key k.
$\overset{k}{\mapsto} R$	k is the public key of principal R.
$S \overset{s}{\leftharpoondown\rightharpoonup} R$	Principals S and R share the secret s.

Table 3.2: Formal security notations

Postulates Table 3.3 itemizes the postulates used in this section.

Postulate name	**Rule**	**Meaning**
Message meaning	$\frac{S \mid\equiv R \overset{K}{\leftrightarrow} S, S \triangleleft \{T\}_K}{S \mid\equiv R \mid\sim T}$	If S believes that K is shared with R, and S sees the encrypted T using K, then S believes that R once said T
Nonce verification	$\frac{S \mid\equiv \#(T), S \mid\equiv R \mid\sim T}{S \mid\equiv R \mid\equiv T}$	If S believes that T is fresh, and believes that R once said T, then S believes that R believes T
Decomposition	$\frac{S \triangleleft (T, U)}{S \triangleleft T}$	If S sees the concatenation (T, U), then it sees T
	$\frac{S \mid\equiv \#(T)}{S \mid\equiv \#(T, U)}$	If S believes that T is fresh, then it believes that (T, U) is fresh
	$\frac{S \mid\equiv (T, U)}{S \mid\equiv T}$	If S believes the concatenation (T, U), then it believes T

Table 3.3: Postulates

Exchanged messages In the realm of message exchange within our protocol, involving the pivotal S and R principals, we observe a sequence of exchanged messages that assume utmost significance during the discovery sub-phase. In this intricate communication process, both A and R engage in the following message exchanges:

$\mathrm{M}_1)$ $S \rightarrow R : \{GUID_S, Nonce_s, MAC(GUID_S || Nonce_s, S_S)\}$

$\mathrm{M}_2)$ $R \rightarrow S : \{GUID_R, Nonce_r, MAC(GUID_R || Nonce_r, S_R)\}$

Idealization The conceptual representations of the aforementioned messages take the following idealized forms:

$\mathrm{I}_1)$ $S \triangleleft \{GUID_R, Nonce_r, \{GUID_R, Nonce_r\}_{S_R}\}$

$\mathrm{I}_2)$ $R \triangleleft \{GUID_S, Nonce_s, \{GUID_S, Nonce_s\}_{S_S}\}$

Goals Our scheme goals of security are as follows:

$G_1)$ $R \mid\equiv S \overset{K_{SB}}{\longleftrightarrow} R$

$G_2)$ $R \mid\equiv S \mid\equiv S \overset{K_{SR}}{\longleftrightarrow} R$

$G_3)$ $S \mid\equiv S \overset{K_{SR}}{\longleftrightarrow} R$

$G_4)$ $S \mid\equiv R \mid\equiv S \overset{K_{SR}}{\longleftrightarrow} R$

Assumptions Our protocol operates under the following assumptions:

$A_1)$ $S \mid\equiv S \overset{S_S}{\rightleftharpoons} R$

$A_2)$ $S \mid\equiv S \overset{S_B}{\rightleftharpoons} R$

$A_3)$ $R \mid\equiv S \overset{S_A}{\rightleftharpoons} R$

$A_4)$ $R \mid\equiv S \overset{S_B}{\rightleftharpoons} R$

$A_5)$ $S \mid\equiv R \mid\equiv S \overset{S_S}{\rightleftharpoons} R$

$A_6)$ $S \mid\equiv R \mid\equiv S \overset{S_R}{\rightleftharpoons} R$

$A_7)$ $R \mid\equiv S \mid\equiv S \overset{S_S}{\rightleftharpoons} R$

$A_8)$ $R \mid\equiv S \mid\equiv S \overset{S_R}{\rightleftharpoons} R$

$A_9)$ $S \mid\equiv \#(Nonce_R)$

$A_{10})$ $S \mid\equiv \#(Nonce_S)$

$A_{11})$ $R \mid\equiv \#(Nonce_S)$

$A_{12})$ $R \mid\equiv \#(Nonce_R)$

Verification and proof Using BAN logic rules and the previously outlined assumptions, we can derive the following formulas:

Applying the decomposition rule to I_1, gives:

F_1)

$$\frac{R \triangleleft \{GUID_S, Nonce_S, \{GUID_S, Nonce_S\}_{S_S}\}}{R \triangleleft (GUID_S, Nonce_S)}$$

F_2)

$$\frac{R \triangleleft \{GUID_S, Nonce_S\{GUID_S, Nonce_A\}_{S_S}\}}{R \triangleleft (\{GUID_S, Nonce_S\}_{S_S})}$$

By applying the message-meaning rule to F_1 and F_2, we can deduce the following:

F_3)

$$\frac{R \mid\equiv S \overset{S_S}{\rightleftharpoons} R, R \triangleleft \{GUID_S, Nonce_S\}_{S_S}}{R \mid\equiv S \mid\sim \{GUID_S, Nonce_S\}}$$

By applying the decomposition rule to A_{10} and A_{11}, we can derive the following:

F_4)

$$\frac{R \mid\equiv \#(Nonce_S)}{R \mid\equiv \#(GUID_S, Nonce_S)}$$

Using F_3 and F_4, we can deduce the following:

F_5)

$$\frac{R \mid\equiv \#(GUID_S, Nonce_S), R \mid\equiv S \mid\sim \{GUID_S, Nonce_S\}}{R \mid\equiv S \mid\equiv \{GUID_S, Nonce_S\}}$$

After discovering that principal S is a neighbor of R, the key agreement sub-phase commences, with R initiating the process by calculating the shared key K_{SR}.

Applying the decomposition rule to assumptions A_3 and A_4 yields:

F_6)

$$\frac{R \mid\equiv S \overset{S_S}{\rightleftharpoons} R, R \mid\equiv S \overset{S_R}{\rightleftharpoons} R}{R \mid\equiv S \overset{K_{SR}}{\longleftrightarrow} R}$$

This concludes the verification of the goal G_1.

By applying the decomposition rule on A_7 and A_8, we obtain:

F_7)

$$\frac{R \mid\equiv S \mid\equiv S \overset{S_S}{\rightleftharpoons} R, R \mid\equiv S \mid\equiv S \overset{S_R}{\rightleftharpoons} R}{R \mid\equiv S \mid\equiv S \overset{K_{SR}}{\longleftrightarrow} R}$$

Which concludes the verification of the goal G_2.

By applying the decomposition rule on I_2, we get:

F_8)

$$\frac{S \triangleleft \{GUID_R, Nonce_R, \{GUID_R, Nonce_R\}_{S_R}\}}{S \triangleleft (GUID_R, Nonce_R)}$$

F_9)

$$\frac{S \triangleleft \{GUID_R, Nonce_R\{GUID_R, Nonce_R\}_{S_R}\}}{S \triangleleft (\{GUID_R, Nonce_R\}_{SR})}$$

By applying the message-meaning rule to formulas F_8 and F_9, it yields:

F_{10})

$$\frac{S \mid\equiv S \overset{S_R}{\rightleftharpoons} R, S \triangleleft \{GUID_R, Nonce_R\}_{SR}}{S \mid\equiv R \mid\sim \{GUID_R, Nonce_R\}}$$

By applying the decomposition rule to assumptions A_9 and A_{12}, we get:

F_{11})

$$\frac{S \mid\equiv \#(Nonce_R)}{S \mid\equiv \#(GUID_R, Nonce_R)}$$

From F_{10} and F_{11}, we conduct:

F_{12})

$$\frac{S \mid\equiv \#(GUID_R, Nonce_R), S \mid\equiv R \mid\sim \{GUID_R, Nonce_R\}}{S \mid\equiv R \mid\equiv \{GUID_R, Nonce_R\}}$$

Upon the discovery of the principal R as a neighboring to S, the initiation of the key agreement sub-phase is undertaken by S, wherein the shared key K_{SR} is derived.

Through the application of the decomposition rule to assumptions A_1 and A_2, we can derive the following:

F_{13})

$$\frac{S \mid\equiv S \overset{S_S}{\rightharpoonup} R, S \mid\equiv S \overset{S_R}{\rightharpoonup} R}{S \mid\equiv S \overset{K_{SR}}{\longleftrightarrow} R}$$

This concludes the verification of the goal G_3.

By applying the decomposition rule to assumptions A_5 and A_6, we obtain:

F_{14})

$$\frac{S \mid\equiv R \mid\equiv S \overset{S_S}{\rightharpoonup} R, S \mid\equiv R \mid\equiv S \overset{S_R}{\rightharpoonup} R}{S \mid\equiv R \mid\equiv S \overset{K_{SR}}{\longleftrightarrow} R}$$

And that concludes the verification of the goal G_4.

Security comparison

Table 3.4 presents a comprehensive security comparison between our scheme and the most recent related works discussed previously, focusing on various well-known attack scenarios.

In our proposed scheme, we achieve a notably high level of resilience against node capture attacks, primarily due to our approach of dividing the network into distinct subsets and subareas. In contrast, other related works such as [45, 46, 1, 47] exhibit lower resilience in this regard. This difference can be attributed to their practice of preloading key materials into the memories of all nodes during the initialization phase, which poses vulnerabilities.

The adaptability of resilience against node capture attacks is a distinctive feature of the scheme presented in [44]. It offers configurability, allowing for high, medium,

or low resilience levels, depending on factors such as network connectivity. However, this flexibility comes with trade-offs. To optimize connectivity and performance, the scheme may preload a larger number of keys per node, resulting in lower resilience against node capture attacks during the initialization phase. This choice also impacts storage requirements.

It's noteworthy that [44] does not address the resilience levels against replay attacks, forward secrecy, and backward secrecy in their evaluation. In contrast, [12] demonstrates high resilience against node capture attacks, primarily because of the centralized nature of smart home systems, where all devices establish direct connections with third-party entities and the home gateway.

Considering a different aspect of security, our scheme demonstrates a heightened resilience against eavesdropping attacks. This enhanced security is attributed to the incorporation of the MAC technique during the pairwise key establishment phase. In this phase, adversaries are limited to intercepting public information, such as the GUID. Additionally, our scheme ensures that no sensitive data is exchanged during this phase; instead, all messages are encrypted at other stages of communication.

In contrast, [45] lacks similar resilience to eavesdropping attacks. This is primarily due to the exchange of group keys in plaintext during the node addition phase, rendering them vulnerable to interception. Similarly, [1] does not offer robust protection against eavesdropping attacks, as it lacks a secure mechanism for updating keys, leaving them exposed to interception in plaintext form.

In terms of resilience against brute force attacks, our scheme offers configurability, thanks to the α security parameter. This parameter can be adjusted to align with the desired security level for the network. On the other hand, the other schemes generally exhibit a relatively high degree of resilience against brute force attacks. However, it's worth noting that the configurability in our scheme comes with certain trade-offs.

One notable trade-off is the impact on storage. For instance, in [45], the scheme preloads all nodes with the complete set of network key materials. While this approach enhances security, it also results in high storage requirements, especially as the network size increases. Similarly, schemes like [47] may experience performance challenges due to the larger size of their one-one functions, which necessitate more extensive computations. Therefore, while our scheme provides flexibility in adapting to different security levels, it's essential to consider the associated storage implications

carefully.

Factorization attacks are another important consideration. Our scheme demonstrates resilience against such attacks, primarily due to its utilization of hash functions for key generation after secret multiplication. This approach enhances security by mitigating vulnerabilities associated with factorization attacks.

However, when examining schemes like [12], they may not exhibit the same level of resilience against factorization attacks. This is because [12] relies on the asymmetric Rabin cryptosystem, which has known vulnerabilities to factorization attacks. In such systems, the private key can potentially be derived by factorizing the corresponding public key, making them susceptible to factorization-based security breaches.

Therefore, the choice of cryptographic mechanisms and algorithms plays a critical role in determining a scheme's resilience against factorization threats, with our scheme's approach of using hash functions contributing to its robustness in this regard.

3.5.2 Performance analysis

The performance evaluation of our scheme involves a comprehensive comparison with recent related works, specifically [45] and [47]. This evaluation focuses on key metrics encompassing communication, computation, storage overheads, and energy consumption. To conduct these assessments, we employ simulations involving nodes situated within a square area of $100m^2$, with node counts ranging from 10 to 100, and each node potentially having 10 to 30 neighbors. The simulation is implemented using Rust-lang [13].

Furthermore, we evaluate the scheme's resilience against node capture attacks through simulations involving 1000 nodes and 100 subsets, with each node having approximately 14 neighbors. This evaluation is carried out using JavaScript [14]. In both simulation scenarios [13, 14], it is assumed that the number of gateways constitutes 10% of the total node count.

In the evaluation process, certain metrics, such as storage, are assessed on an individual node basis, while others, like resilience against node capture attacks, are

Scheme	Year	F1	F2	F3	F4	F5	F6	F7	F8
EVKMS	_	HIGH	✓	✓	✓	✓(Configurable)	✓	✓	✓
Nafi et al. [45]	2020	LOW	✓	✓	✗(Group key in node addition)	✓	✓	✓	✓
Nafi et al. [46]	2021	LOW	✓	✓	✓	✓	✓	✓	✓
H2KD [1]	2022	LOW	✓	✓	✗(Key update)	✓	✓	✓	✓
Bai et al. [12]	2022	HIGH	✓	✓	✓	✓	✗	✓	✓
IFKMS [47]	2022	LOW	✓	✓	✓	✓	✓	✓	
Msolli et al. [44]	2023	LOW	✓	_	✓	✓	✓	_	_

F1: Initialization node capture, F2: Node capture, F3: Replay, F4: Eavesdropping, F5: Brute force, F6: Factorization, F7: Forward secrecy, F8: Backward secrecy

Table 3.4: Comparative study and security evaluation against some known attacks

evaluated at the network level. The results are derived from averaging the outcomes of 1000 simulation runs. This averaging approach is employed to mitigate the impact of random variations and provide a more reliable estimation of the system's overall performance characteristics.

In our simulations, we utilize *mica*2*dot* motes, which are equipped with an *Atmega*128 microcontroller. The selection of the *Atmega*128 microcontroller is motivated by its attributes, including a low-power $8-bit$ architecture, $128KB$ flash memory, $16KB$ SRAM, $4KB$ EEPROM, 54 I/O pins, a clock speed of $16MHz$, and an operating voltage of $3.3V$ [1]. These specifications render it highly resource-constrained, making it suitable for testing the efficiency of our protocol.

Moreover, we propose enhancements aimed at reducing energy consumption by conducting comparative assessments of certain operations with respect to an *Atmega*328*P* microcontroller [2].

We have provided a comparative analysis in Table 3.5, evaluating the performance of our scheme against the key management protocols discussed in the related work section. The table offers insights into the relative performance of each scheme regarding memory utilization, communication overhead, and computational complexity during the key establishment phase.

It's important to highlight some key aspects of the table's content:

- Storage overhead reflects the schemes' memory consumption immediately after establishing the pairwise keys and storing them in the nodes' memory.
- Communication overhead values represent the load on a node within the schemes during the pairwise key establishment phase. These values are assessed by considering the size of messages sent and received among constrained devices across the entire network.
- Computation overhead metrics gauge the computational burden placed on a node within the schemes during the pairwise key establishment phase. This assessment considers the number of multiplications and additions performed by a node when establishing pairwise keys between just two nodes.

[1] `http://www.cmt-gmbh.de/Mica2dot.pdf`

[2] `https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega Datasheet.pdf`

- Additionally, it's worth noting that the storage overhead estimations for all the compared schemes do not consider the established keys.

Within our scheme, the storage overhead varies among nodes. Gateway nodes, for instance, store a maximum of 3 identifiers from their adjacent subsets, in addition to $d+1$ nodes' secrets, each associated with its respective local identifier. This explains why they store $d+1$ elements instead of just d. Conversely, constrained nodes only need to store their own secrets and local identifiers, resulting in a smaller storage footprint.

In contrast, in the [47] scheme, nodes are required to store $d+1$ nodes' identifiers and $d+1$ one-one functions, along with the inverse of their own. Meanwhile, in [45], each node must store its required secrets in the form of a $(d+1)(d+1)$ matrix. This is due to the fact that $d*d$ of these secrets belong to neighboring nodes, while a row and a column, totaling $2d+1$, contain the secrets of the node itself. [44] mandates storing $m*(l_{id}+l_k)$, where m represents the key ring size.

Concerning communication overhead, both our scheme and [45] only require the exchange of 2 messages between two nodes to establish a pairwise key. Conversely, [47] necessitates 3 messages, where 2 of them contain the same content as the other two schemes (message type, node identifier, nonce, and a MAC), while the third involves the encryption of two concatenated nonce values. On the other hand, [44] mandates sending the identifiers of all its keys, equivalent to $m*l_{id}$ for each node.

In terms of computation overhead, our scheme involves only 1 multiplication of the two nodes' secrets and 1 concatenation operation to obtain its GUID. In contrast, [45] requires 2 multiplications and 1 subtraction to calculate the determinant of the $2*2$ nodes' secrets matrix. Conversely, [47] involves calculating the result of 2 one-one functions, performing 3 hash operations, and executing 1 encryption operation. In the case of [44], it requires $m*log(m)$ comparisons, 2 hash operations, and one encryption operation.

Resilience to node capture attack

In the context of our scheme, three potential scenarios arise when considering resilience to node capture attacks.

Scheme	Year	Memory	Communication	Computation
Blom [16]	1984	$2(\lambda+1)*l_{me}$	$\lambda+1$	$\lambda+1$ multiplications & λ additions
Du et al. [22]	2005	$((\lambda+1)\tau+1)l_{me}$	τ indexes & The seed of the column of G or the node id	$2\lambda+1$ multiplications & λ additions
Yu et al. [67]	2008	$(2+\omega)(\lambda+1)l_{me}$	$\lambda+1$	$\lambda+1$ multiplications & λ additions
Zhang et al. [69]	2018	$(\lambda+1)l_{me}$	request and $row_i(A_c)$	$2\lambda+1$ multiplications & λ additions
Nafi et al. [45]	2020	$(d+1)(d+1)l_s$	2 Hello messages	2 multiplications & 1 subtraction & MAC
Nafi et al. [46]	2021	$(d+1)*log(q)$	2 Hello messages	2 evaluated polynomial terms & generation of a random key
H2KD [1]	2022	$l_{id}+l_k$	$3*l_{id}+l_e$	$-$
Bai et al. [12]	2022	$l_{id}+l_k$	$6*l_{id}+2*l_h$	$3*XOR+H$
IFKMS [47]	2022	$(d+1)*l_{id}+(d+2)*l_f$	$2(l_t+l_{id}+l_{of}+l_h)+l_e$	$2F+3H+E$
Msolli et al. [44]	2023	$m*(l_{id+l_k})$	$m*l_{id}$, $m*logMbits$	$m*log(m)$ comparisons & $2H+E$
EVKMS	$-$	Gateway: $3*l_{gid}+(d+1)*(l_s+l_{id})$ Constrained: $3*l_{gid}+l_s+(d+1)*l_{id}$	2 Hello messages	1 multiplication & 1 concatenation & MAC

Table 3.5: Comparative study of the literature review schemes' performance

First, let us consider the situation where the adversary captures a gateway node. In this case, the adversary's ability to decrypt messages from other nodes is severely limited due to the protection provided by TPM [36, 8] employed in gateway nodes. Similarly, in the [45] scheme, the presence of TPMs ensures resilience to gateway node capture attacks. However, the [47] scheme is vulnerable in this regard, as the adversary can access all the information stored in the memory of a captured gateway node.

Now, let us explore the second scenario where the adversary captures a constrained node at the early stages of network deployment when nodes still retain the preloaded vectors and sensitive materials in their memories. In this situation, the adversary can decrypt, at most, the links connecting the captured node's group with nodes in the adjacent groups. However, the adversary cannot compromise the links involving nodes in other groups. In contrast, both the [45] and [47] schemes face a more severe vulnerability. If the adversary captures a single constrained node, they can compromise all network links because these schemes preload all the network key materials and sensitive data on each of their nodes' memories.

The results of our simulations, based on 1000 runs, were used to determine the fraction of compromised links when considering the capture of 1 to 20 nodes. These results are presented in Figure 3.10, illustrating the varying degrees of resilience in different schemes.

In the [47] scheme, the adversary can compromise all network links if they manage to capture at least one node, regardless of its type. This severe vulnerability stems from the fact that all nodes store key materials in their memories without any protection.

On the other hand, the [45] scheme provides better protection. In this scheme, the adversary can only compromise the links shared by the constrained nodes, as TPMs safeguard the memories of gateway nodes. However, this protection is limited to links involving constrained nodes.

In contrast, our scheme offers significantly higher resilience against node capture attacks. When the adversary captures nodes, they can only compromise the links shared by the constrained nodes within the same subset as the captured nodes and those belonging to adjacent subsets. Importantly, our scheme ensures that even gateway nodes' memories are protected by TPMs.

To quantify the difference in resilience, when capturing 20 nodes, our scheme proves to be up to 96.43% more resilient compared to both the [45] and [47] schemes.

In the third scenario, the adversary captures one of the constrained nodes after erasing their initialization information from their memories, which occurs after the pairwise and group-wise key establishment phases. In this case, the adversary can decrypt only the messages can be exchanged among the neighbors of the captured node. As a result, the adversary compromises a total of d bidirectional links, corresponding to the connections between the captured node and its one-hop neighbors.

In the [45] scheme, the adversary's capabilities extend further. They can compromise not only the links between the captured node and its one-hop neighbors but also the links among the neighbors themselves. This expanded vulnerability arises because the adversary gains access to the keys of the captured node and the secrets of the other nodes in its vicinity, allowing them to compromise the links involving these nodes as well.

In contrast, the [47] scheme offers a level of protection by only enabling the adversary to compromise the links between the captured node and its immediate neighbors, without affecting the links among the neighbors themselves.

Following the execution of 1000 simulations, we analyzed the fraction of compromised links for various numbers of compromised nodes, as detailed in the resilience simulation study [14]. These findings have been visually presented in Figure 3.11.

The results clearly demonstrate that our scheme exhibits superior resilience against this type of attack in comparison to other schemes. Specifically, our scheme outperforms [47] by a margin of 9.89% and surpasses [45] by an even greater margin of 54.69%. This showcases the effectiveness of our security measures in mitigating the impact of node capture attacks on network integrity.

Storage overhead

Regarding the storage overhead in our scheme, we need to store three vectors, each containing m elements representing the preloaded secrets of nodes. Additionally, we store the pairwise keys of neighbor nodes in the form of a vector with a length of d,

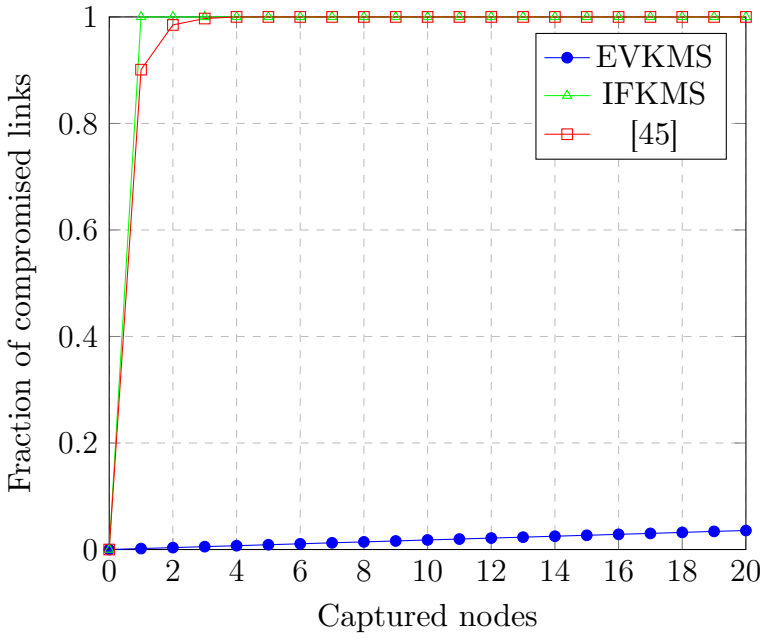


Figure 3.10: Fraction of compromised links in the case of a node capture attack before erasing the initialization sensitive data from the memory of nodes

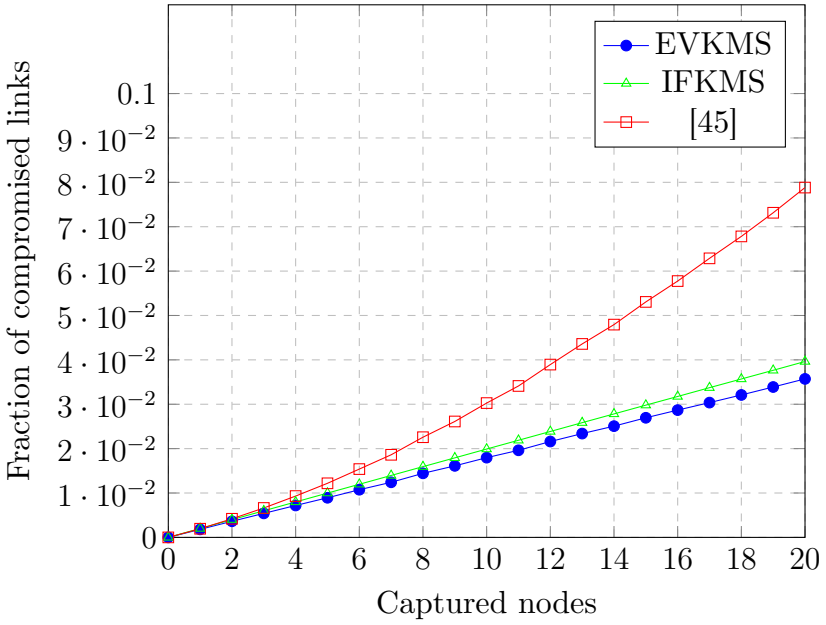


Figure 3.11: Fraction of compromised links in the case of a node capture attack after erasing the initialization sensitive data from the memory of nodes

where d represents the number of node neighbors.

Assuming that the group id size, l_{gid}, is 2 bytes, the node id size, l_{id}, is 2 bytes,

the pairwise key size is 16 bytes, and the size of each neighbor secret is 4 bytes, we can perform the following calculations:

Each node has an average of 14 neighbors, as per [21]. In our case, we assume each node has 15 neighbors, with 5 in each group. In the worst case, our scheme stores $d+1$ node identifiers. With these assumptions, the storage overhead of our scheme is calculated as follows: $3*gid+ls+d*l_k+(d+1)*l_{id} = 3*2+4+15*16+16*2 = 282$ bytes

This storage overhead is significantly lower and negligible when compared to the storage capacity of sensor devices.

In contrast, other schemes, especially matrix-based ones, have much higher storage overhead. Taking Nafi et al.'s matrix-based scheme [45] as an example, it needs to store $(d+1)(d+1)+d$ elements in memory. This results in a storage overhead of: $(d+1)^2*l_s+d*l_k = 256*4+15*16 = 1264$ bytes, which is considerably higher than in our scheme.

For [47], the storage overhead is calculated as follows: $(d+4)*l_{id}+(d+2)*lf+d*l_k = (15+4)*4+(15+2)*16+15*16 = 560$ bytes, which is also higher than in our scheme.

Figure 3.12 provides a visual comparison of the storage overhead between our scheme, [45], and [47]. This comparison considers the constrained devices' storage overhead in our proposal and utilizes the same constraints and parameters for all the other proposed schemes in the figure.

During the deployment phase of our scheme, the storage requirements entail maintaining the preloaded secrets of each node. Specifically, we need to manage three distinct vectors denoted as V_p, V_c, and V_n as previously explained, each associated with a node's preloaded secrets. It is important to note that the size of these vectors, denoted as x, can vary for nodes, as the number of nodes in each subset may differ.

Additionally, we are tasked with storing three group identifiers, one for each subset of the network. Consequently, the storage overhead during the deployment phase in our scheme can be expressed as $3*l_{gid}+3*x*l_s$. By plugging in the values, this leads to a storage overhead of $32+3x4 = 6+12x$ bytes. This comprehensive storage allocation accounts for the varying sizes of vectors and ensures efficient storage management

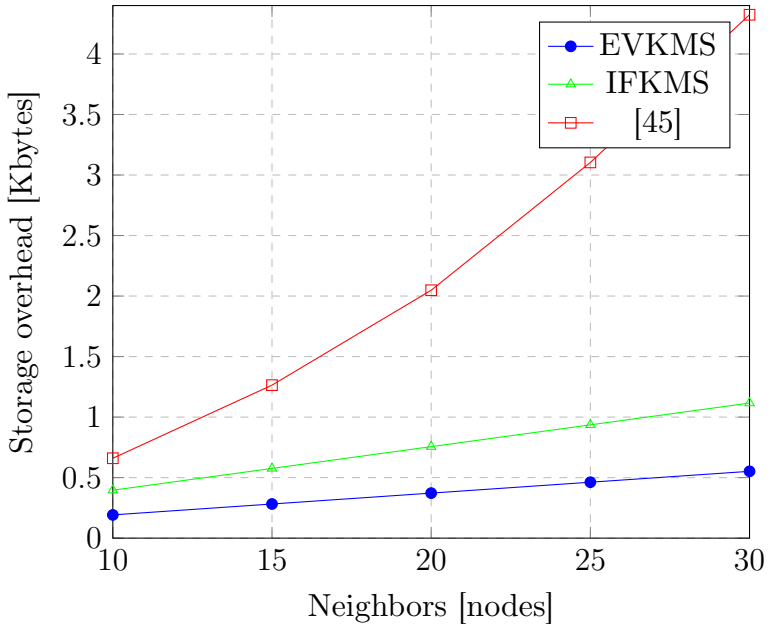


Figure 3.12: Storage overhead comparison during the pairwise key establishment phase

during the deployment phase.

In contrast to other schemes, our scheme demonstrates significantly lower storage overhead during the deployment phase. Let's consider two prominent examples for comparison:

- In the scheme presented by Nafi et al. [45], a constrained node is required to store an entire matrix with dimensions $n \times n$, where n represents the total number of nodes in the network. Consequently, the storage overhead for this scheme becomes $n \times n \times l_s$, where l_s is the size of an individual secret. In essence, this leads to a storage requirement of $(n^2) \times 4 = (2n)^2$ bytes, a considerably higher figure when compared to our scheme.
- In the context of the [47] scheme, the storage overhead for necessary preloaded data amounts to $n \times l_{id} + (n+1) \times l_f$. This results in a storage overhead of $4n + 16(n+1) = 20n + 16$ bytes, once again surpassing the efficiency of our scheme.

During the deployment phase, Figure 3.13 visually compares the storage overhead of our scheme with those in [45] and [47]. Notably, the storage overhead analysis

focuses on a single network node, considering two variables: network size and subset sizes, instead of fixing one while varying the other. The figure clearly illustrates that our scheme consistently maintains the lowest storage overhead during deployment.

Both alternative schemes exhibit varying storage overheads depending on the total number of nodes in the network. While [45] demands the storage of an entire matrix, which can become substantial, [47] requires a linear number of key materials. In contrast, our scheme adapts its storage overhead requirements based on the number of nodes within the same and adjacent subsets of the current node. This dynamic approach allows our scheme to efficiently handle massive networks, achieving storage savings of up to 81.84%, particularly as the network size expands to include 100 nodes, with 30 nodes in each subset.

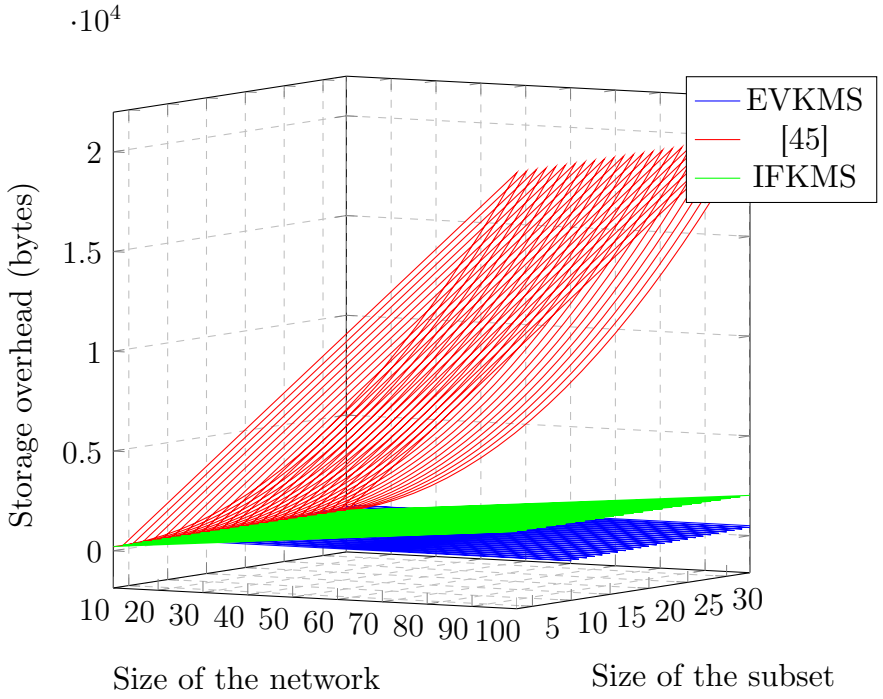


Figure 3.13: Comparison of storage overhead of different schemes during the deployment phase according to the network and the subset sizes

Communication overhead

The communication cost is intricately linked to the number of exchanged messages and their respective sizes. In Table 3.6, we conduct a comprehensive evaluation of communication costs within each phase of our scheme. This evaluation includes a breakdown of sent and received messages, their sizes, and the associated energy cost,

all segmented by node type. It's crucial to recognize that communication costs may vary according to node type in certain scenarios. For example, during the initialization phase, no mandatory messages need to be exchanged, resulting in negligible communication energy costs. Conversely, during the pairwise key establishment phase, each node must send a broadcast message and receive d acknowledgment (ACK) messages from its neighboring nodes. The size of each message includes parameters such as the message type, sender node's GUID, and nonce, among others.

For our evaluation, we assume specific parameters: $10 \leq d \leq 30$, $l_{type} = 1$, $l_{GUID} = 4$, $l_{nonce} = 4$, $16 \leq l_{MAC} \leq 64$, and $m = 58$, with the unit of size being bytes. Additionally, we assume that encryption employs a block cipher algorithm, which outputs one or more blocks for [47] and [44].

Figure 3.14 provides the results of simulations [13] illustrating the communication overhead of our scheme, [45], [47], and [44]. The evaluation is conducted concerning variables such as the number of neighbors, MAC size, and encryption block size, all within the context of the pairwise key establishment phase. Equation 3.21 illustrates how the encryption block size can impact the communication overhead for [45] and [44] schemes.

$$l_{message} = \begin{cases} l_{message} & \text{if } l_{message} \mod l_{block} = 0 \\ \left(\left\lfloor \frac{l_{message}}{l_{block}} \right\rfloor + 1 \right) * l_{block} & \text{otherwise} \end{cases} \tag{3.21}$$

The findings presented in this figure demonstrate that our scheme exhibits a slight advantage over the [47] and [44] schemes in terms of communication overhead. This advantage can be attributed to specific factors within each scheme's design.

In the case of [47], it requires an additional encrypted message exchange compared to our scheme, as detailed in Table 3.5. This additional message contributes to a marginally higher communication cost for [47].

On the other hand, the communication overhead of [44] is directly influenced by the encryption block size, as elucidated in Equation 3.21. This dependence on block size introduces some variability in the communication overhead for [44], leading to fluctuations.

In contrast, the [45] scheme necessitates the exchange of messages of identical size

as our scheme. Therefore, its communication overhead aligns closely with that of our scheme.

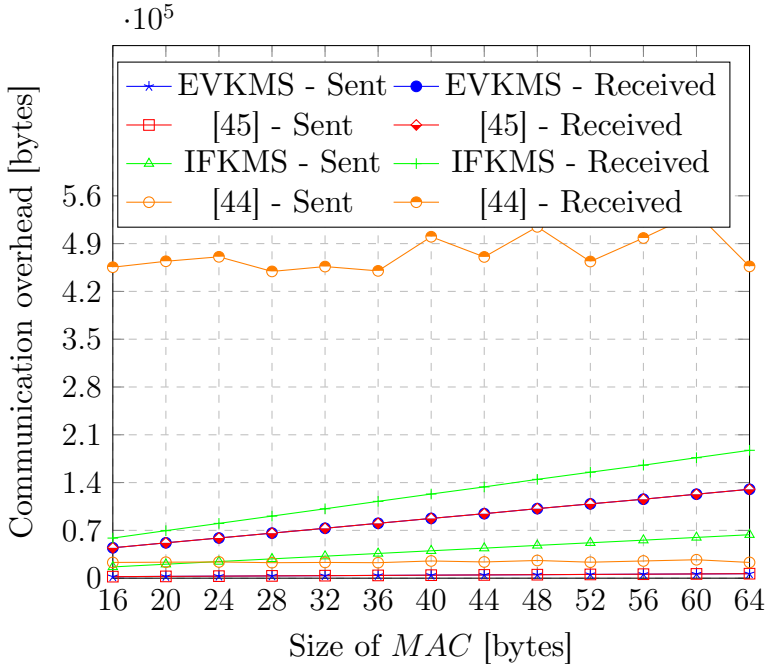


Figure 3.14: Pairwise key establishment communication overhead comparison ($m = 58$)

In the group-wise key establishment phase, our scheme distinguishes itself by not incurring any communication overhead. The group-wise key is exclusively established through computational operations utilizing the preloaded secrets of the nodes.

Contrastingly, other schemes like [45] necessitate communication overhead for group-wise key establishment. In [45], gateways are required to send and receive messages, including ACK messages, totaling d messages for each gateway. Constrained nodes in this scheme also contribute to the communication overhead by receiving at least one message and sending one ACK message for each constrained node.

To provide a clearer perspective on this distinction, Figure 3.15 illustrates the communication overhead comparison between our scheme and [45] for various neighbor counts during the group-wise key establishment phase. Notably, our scheme's communication overhead remains at zero, as it relies on local computation facilitated by the preloaded subset secrets of the nodes. In contrast, [45] experiences higher communication overhead, particularly for gateway nodes, due to the need to exchange multiple messages. This discrepancy in communication overhead is evident in the figure.

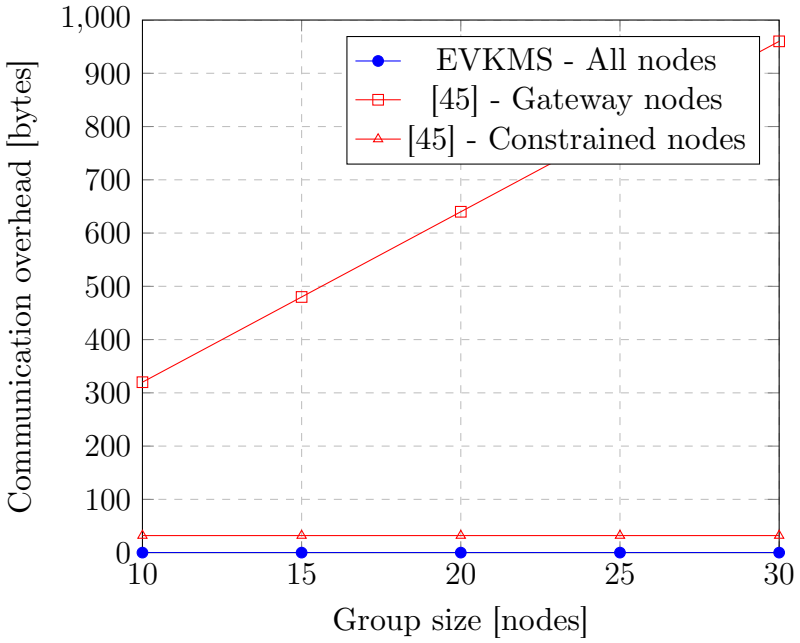


Figure 3.15: Group-wise key establishment communication overhead comparison

Computation overhead

In our scheme, establishing pairwise keys with neighbor nodes involves a single multiplication and concatenation operation for each key, resulting in a total of d multiplications and d concatenations to establish all pairwise keys.

In contrast, [45] requires each node to perform $2d$ multiplications and d additions to establish all pairwise keys with its neighbors.

To evaluate the computational costs associated with different operations such as hashing, encryption, and decryption, we reference measurements provided in [65] and [35]. According to [65], the energy consumption for $AES-128$ encryption and decryption operations, including key setup, is $1.62\mu J$ and $2.49\mu J$ per byte, respectively. Additionally, the energy consumption for the Secure Hash Algorithm 1 (SHA-1) is $5.9\mu J$ per byte. These measurements are based on the $mica2dot$ device, which features an $Atmega128$ microcontroller. Measurements in [35] encompass various microcontrollers, including the $Atmega328P$.

Table 3.7 provides an overview of the computation overhead in our scheme during both the pairwise key establishment and group-wise key establishment phases. As depicted in the table, our scheme exhibits slightly lower computational overhead

Phase	**Node type**	**Sent messages**	**Received messages**	**Sent bytes**	**Received bytes**	**Energy**
Initialization	All	0	0	0	0	0
Pairwise key establishment	All	1	d	$l_{type} + lnonce + l_{gid} + l_{id} + l_{mac}$	$d(l_{type} + lnonce + l_{gid} + l_{id} + l_{mac})$	$(l_{type} + lnonce + l_{gid} + l_{id} + l_{mac}) * EPSB + d(l_{type} + lnonce + l_{gid} + l_{id} + l_{mac}) * EPRB$
Group key establishment	All	0	0	0	0	0
New node addition	Gateway	3	$d+1$	$3E$	$(d+1)E$	$3E * EPSB + E * (d+1) * EPRB$
	New node	2	d	$E + l_{type} + lnonce + l_{gid} + l_{id} + l_{mac}$	$E + (d-1)(l_{type} + lnonce + l_{gid} + l_{id} + l_{mac})$	$(E + l_{type} + lnonce + l_{gid} + l_{id} + l_{mac}) * EPSB + (E + (d-1)(l_{type} + lnonce + l_{gid} + l_{id} + l_{mac})) * EPRB$
	Simple node	2	2	$E + l_{type} + lnonce + l_{gid} + l_{id} + l_{mac}$	$E + l_{type} + lnonce + l_{gid} + l_{id} + l_{mac}$	$(E + l_{type} + lnonce + l_{gid} + l_{id} + l_{mac}) * EPSB + (E + type + lnonce + l_{gid} + l_{id} + l_{mac}) * EPRB$
Node revocation	Revoked node	1	d	E	$d * E$	$E * (EPSB + d * EPRB)$
	other nodes	1	1	E	E	$E(EPSB + EPRB)$
New node joining key refresh	All	1	d	E	$d * E$	$E * (EPSB + EPRB)$
Node leaving key refresh	Gateway	d	d	$d * E$	$d * E$	$(d * E)(EPSB + EPRB)$
	Simple node	1	1	E	E	$E * (EPSB + EPRB)$
Periodic key refresh	All	0	0	0	0	0
Key refresh request	Gateway	1	d	E	$d * E$	$E * (EPSB + (d * EPRB))$
	Simple node	1	1	E	E	$E * (EPSB + EPRB)$

Table 3.6: Analysis of communication costs per phase

compared to [45] and other related schemes. Notably, there is potential to further optimize our scheme's computational overhead by employing more efficient algorithms such as $ChaCha20$ instead of $AES-128$ and $BLAKE2$ instead of SHA-1, which have demonstrated lower energy consumption based on measurements in [35] and [3], respectively.

Phase	**Encryptions**	**Decryptions**	**Hashes**	**Energy** (μJ)
Pairwise key establishment	0	0	$2d$	$2d * (5.9)$
Group-wise key establishment	1	0	0	1.62

Table 3.7: EVKMS computation costs during the pairwise and group-wise key establishment phases.

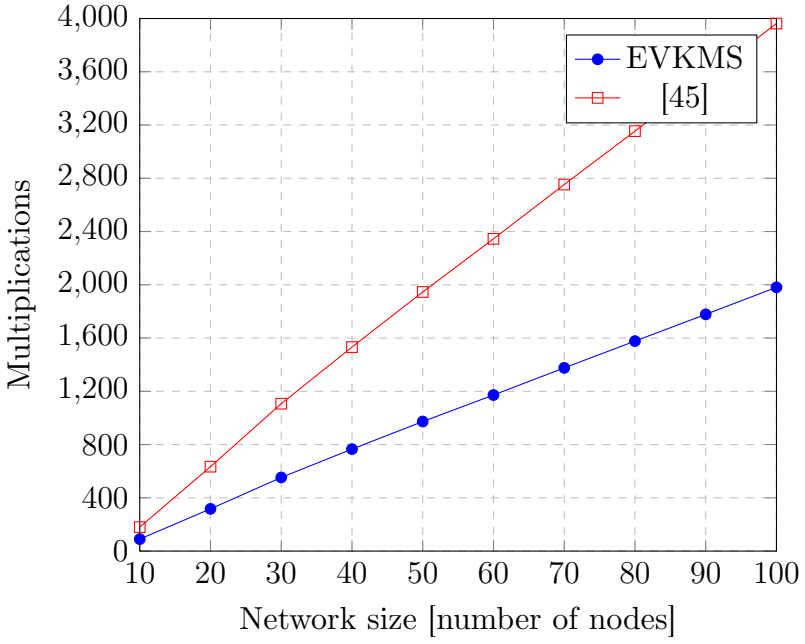


Figure 3.16: Number of multiplications according to the network size during the pairwise key establishment phase.

In relation to the network size, Figure 3.16 offers a graphical depiction of the multiplication counts needed in the pairwise key establishment phase. This graph presents simulation results for both [45] and our scheme, with [47] and [44] excluded due to their varying numbers of multiplications, which can be dependent on specific factors.

The results clearly demonstrate that our scheme outperforms [45] in terms of computational efficiency. Specifically, our scheme achieves a reduction of 50% in the

number of required multiplications compared to [45]. This efficiency gain is a significant advantage, especially as the network size grows, as it reduces computational overhead and contributes to improved performance.

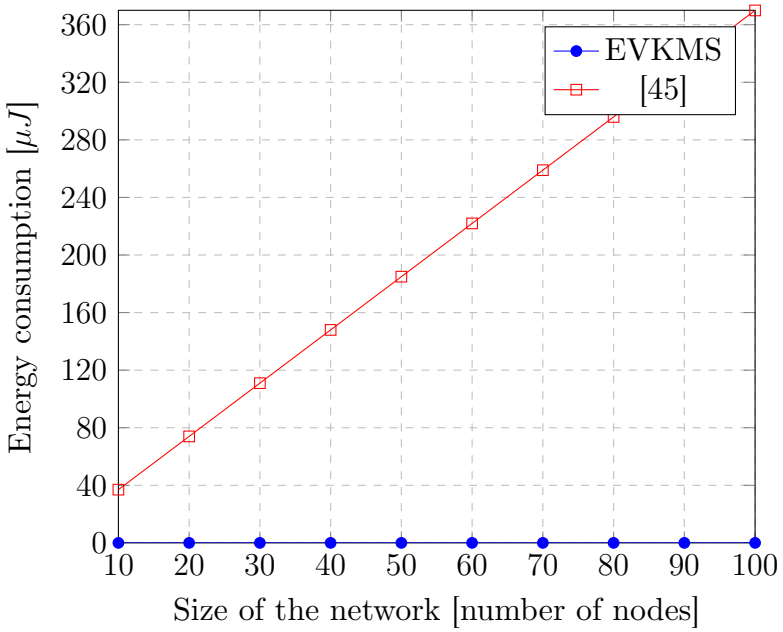


Figure 3.17: Computation energy consumption according to the network size during the group-wise key establishment phase.

Figure 3.17 illustrates the results of a simulation analyzing computational energy consumption during the group-wise key establishment phase, with a focus on constrained nodes. This simulation involved [45] and our scheme, considering a message size of 16 bytes for [45] and the highest security level for our scheme, which employs hash operations rather than encryption.

The outcomes clearly showcase the significant energy efficiency of our scheme compared to [45]. In our scheme, constrained nodes perform only a single hash operation to establish a group-wise key. In contrast, [45] requires constrained nodes to undertake more complex operations, including receiving encrypted keys, decryption, and encryption for ACK messages. As a result, our scheme exhibits substantial energy savings, reaching up to 99.99% when considering a network of 100 nodes compared to [45].

It's worth noting that these results reflect the worst-case scenario for our scheme and the best-case scenario for [45], as using key lengths of less than 16 bytes could pose security risks for [45].

Energy consumption

The energy consumption for communications in various phases of our scheme is influenced by several factors, including the number of neighbors of the node, the hash size, and the size of encrypted data, as outlined in Table 3.6.

The size of encrypted data generated by Advanced Encryption Standard (AES) encryption depends on both the input data length and the chosen mode of operation. For instance, when encrypting 16 bytes of plaintext data using $AES{-}128$ in Electronic Code Book (ECB) or Cipher Block Chaining (CBC) mode, the size of the encrypted data remains at 16 bytes (equivalent to one block of 16 bytes). In these cases, each plaintext block is independently encrypted, and the plaintext is padded to a multiple of the block size (16 bytes) before undergoing encryption. Additionally, each block is XORed with the previous ciphertext block before encryption, following the AES encryption standard [23].

The evaluation of energy consumption for our protocol and related protocols occurs within the same environment and setup used for assessing communication overhead. Energy consumption in our protocol and others is directly tied to the number of messages exchanged and their sizes. Thus, energy consumption increases as these factors increase and decreases as they decrease. According to measurements from [65], the energy cost of sending one byte from a mica2dot device is $EPSB = 59.2\mu J$, while the energy cost for receiving one byte is $EPRB = 28.6\mu J$.

Figure 3.18 offers a comparative view of the communication energy consumption for our scheme as well as the [45], [47], and [44] schemes during the pairwise key establishment phase. Energy consumption is assessed concerning MAC and encryption block sizes, with the number of neighbors varying from 10 to 30. In the case of [44], we consider a keyring size of $m = 58$ as the probability of sharing at least one key is high when $|S| = 1000$.

As evident from the figure, our scheme consumes a comparable amount of communication energy to the [45] scheme, which is calculated as $d * (l_{type} + l_{id} + l_{nonce} + l_{mac}) * EPRB$ for received messages and $(l_{type} + l_{id} + l_{nonce} + l_{mac}) * EPSB$ for sent messages. However, our scheme performs fewer computation operations, requiring only one multiplication and one concatenation. In contrast, [45] needs two multiplications and one subtraction, resulting in lower energy consumption for our scheme.

Additionally, the [47] scheme consumes more communication energy than our scheme due to the additional encrypted message. On the other hand, [44] requires more energy than the other schemes, and its energy consumption fluctuates. This fluctuation is attributed to the nature of the Shift-AES block cipher encryption algorithm, as indicated in equation 3.22.

$$l_{message} = \begin{cases} m * l_{id} & \text{if } (m * l_{id}) \mod l_{block} = 0 \\ \left(\left\lfloor \frac{m * l_{id}}{l_{block}} \right\rfloor + 1 \right) * l_{block} & \text{otherwise} \end{cases} \tag{3.22}$$

The comparison of energy consumption for group-wise key establishment communications is illustrated in Figure 3.19. In this analysis, energy consumption is determined based on the size of exchanged messages during the group-wise key establishment phase. Specifically, we assume a fixed size of encrypted data, denoted as $E = 16$ bytes while varying the number of neighbors from $d = 10$ to $d = 30$.

Notably, our scheme demonstrates minimal energy consumption for the required communications during the group-wise key establishment phase. Unlike our scheme, the [45] scheme incurs energy consumption because it needs to exchange messages. In [45], each gateway node must send and receive d messages, and each constrained node must send and receive at least one message, as depicted in Figure 3.15. These message exchanges result in increased energy consumption for the [45] scheme compared to our approach, where no such exchanges are required.

Figure 3.20 provides an overview of the total energy consumption concerning network size during the group-wise key establishment phase, considering only the energy consumption for all constrained nodes [13]. This comprehensive analysis combines the results of communication and energy consumption.

Our scheme stands out as more energy-efficient compared to the scheme proposed in [45]. The key reason behind this efficiency is that our scheme eliminates the need for communication during this phase, resulting in no communication energy consumption, as illustrated in Figure 3.19. Additionally, our scheme outperforms in terms of computation efficiency, as shown in Figure 3.17.

When benchmarked against the schemes outlined in [47] and [44], our approach yields energy savings of up to 99.99% in a network with 100 nodes. This substantial reduction in energy consumption attests to the heightened efficiency of our scheme

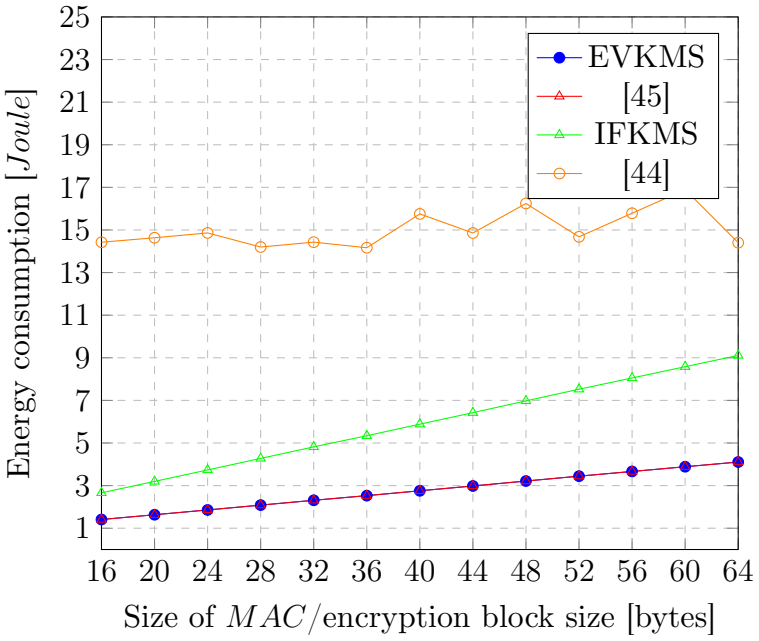


Figure 3.18: Communication energy consumption comparison during pairwise key establishment comparison ($m = 58$)

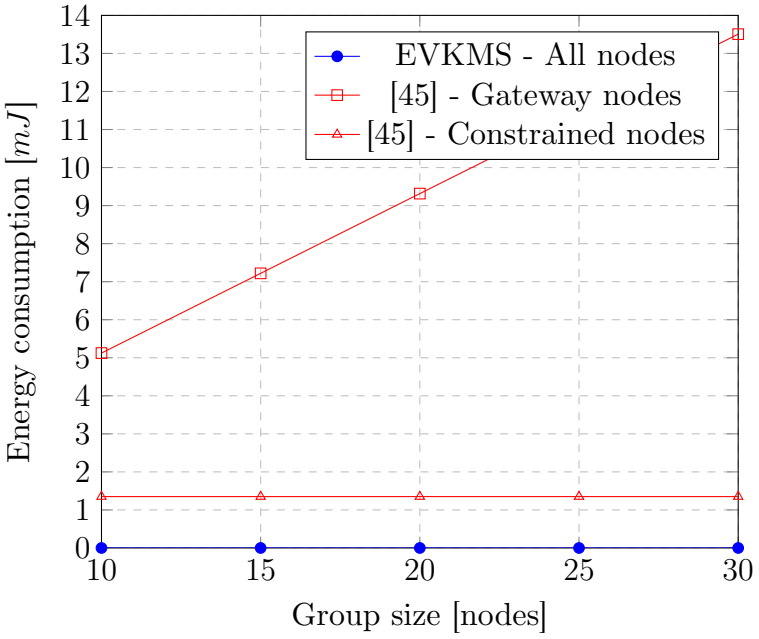


Figure 3.19: Communication energy consumption during group-wise key establishment comparison

in establishing group-wise keys.

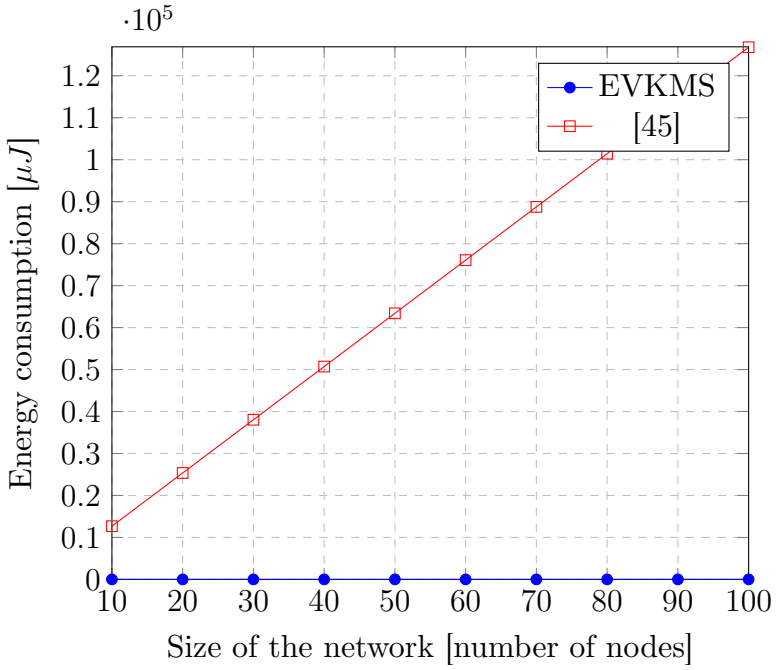


Figure 3.20: Total energy consumption according to the network size during the group-wise key establishment phase.

3.6 Conclusion

In conclusion, this chapter has provided an extensive exploration of EVKMS, a key management scheme designed to bolster the security of communications within IoT networks. EVKMS employs innovative techniques, including the use of pre-distributed vectors and area division, to safeguard cryptographic keys and sensitive data during transmission over insecure channels. It demonstrates scalability, support for new nodes, and efficient operation without imposing excessive computational demands.

Our rigorous performance evaluation has demonstrated acevkms's superior performance compared to existing schemes, offering promising solutions to the intricate security challenges faced by IoT networks.

Looking ahead, it is crucial to acknowledge that while our simulations have showcased EVKMS's commendable performance, they may not capture the nuances of computational overhead accurately. Complex operations such as multiplication warrant further investigation through real-world experiments, which constitute a promising avenue for future research.

With the groundwork laid by EVKMS, we now transition to the subsequent chapter, which introduces another pioneering contribution: BKRSC-IoT. This innovative approach harnesses smart contracts, along with blockchain technology to address the critical issue of key revocation within IoT networks, ushering in a new era of security and performance.

Contribution: Blockchain-based Key Revocation using Smart Contracts for IoT networks

4.1 Introduction

The advent of the IoT has ushered in a transformative era in human interaction with technology, permeating domains like smart homes, wearable devices, and industrial automation. However, this surge in interconnected devices has brought forth significant security and privacy concerns. Among these challenges, effective key management for resource-constrained IoT devices stands as a critical security issue. Within this context, the process of key revocation emerges as a pivotal phase in key management schemes.

Traditional systems have historically relied upon centralized authorities to oversee key revocation, a practice fraught with vulnerabilities, single points of failure, and substantial communication overhead. Centralized systems risk systemic disruption if the central authority is compromised or becomes unavailable, thereby exposing the entire system to potential security breaches. Moreover, key revocation in centralized systems often incurs high energy consumption and communication overhead due to the constant need for devices to interact with the central authority to manage their cryptographic keys.

Blockchain, originally conceived as a decentralized and distributed ledger technology, has garnered attention for its ability to provide a trustless and tamper-proof environment, applicable to a wide array of use cases. Its inherent attributes, including decentralization, immutability, and security, position it as a promising solution for key management within IoT networks, particularly in the context of key revocation.

This chapter introduces BKRSC-IoT a pioneering solution that harnesses blockchain technology to decentralize the key revocation phase within IoT networks, showcasing the feasibility and effectiveness of employing blockchain-based smart contracts for key revocation in resource-constrained IoT devices. Additionally, this chapter aims to conduct an in-depth analysis of the potential advantages inherent in blockchain-based key revocation within IoT networks, including heightened security, reduced communication overhead, and improved energy efficiency.

The subsequent portions of this chapter thoughtfully explore BKRSC-IoT, covering aspects such as its background, smart contract design and implementation, performance and security assessment, and, finally, a conclusion that encapsulates key findings and outlines future research directions in this evolving field.

4.2 Preliminaries

Blockchain, aptly named for its composition of interlinked blocks, serves as a decentralized ledger technology. Its inception can be traced to 2008, marked by the seminal publication of a paper by Satoshi Nakamoto detailing decentralized, peer-to-peer cash transactions [48]. The core tenet of Blockchain is the establishment of a distributed ledger, shared among all network participants. This ledger, housing an exhaustive account of network transactions, is disseminated across all participants and harmonized via a consensus algorithm—a protocol designed to engender agreement among these stakeholders. Blockchain confers the advantages of distribution, security, traceability, and immutability [41, 64].

The literature offers a spectrum of consensus algorithms designed to address the challenge of achieving a shared consensus among blockchain participants. Prominent among these are Proof of Work (PoW) [48], Proof of Stake (PoS) [60], Delegated Proof of Stake (DPoS) [37], and Practical Byzantine Fault Tolerance (PBFT) [20]. These algorithms assume a pivotal role in ensuring the security of the Blockchain.

Blockchain can be categorized into three distinct types: public, private, and consortium [68]. In public Blockchain, unrestricted participation is extended, enabling anyone to join the network and engage in the consensus algorithm. In contrast, private Blockchain confines participation to a predefined set of participants within the same organization. Consortium Blockchains strike a balance by allowing participants from diverse organizations to partake in the consensus algorithm. The distinctive merits of private and consortium Blockchains lie in their agility and energy efficiency relative to public Blockchains. However, they are constrained in terms of participant inclusivity compared to public Blockchains.

Smart contracts, a concept originating from Nick Szabo in 1994 [57], represent self-executing programs within the Blockchain. These contracts automate the execution of agreements between two or more parties, thus mitigating costs and time expenditures without necessitating the involvement of a trusted intermediary.

The applications of Blockchain span various domains, offering solutions to challenges related to key management and data integrity, as exemplified in works such as those by Ma et al. [41], Lei et al. [38], Kandi et al. [34], and Hameedi et al. [32].

4.3 Proposed solution

4.3.1 Network model

Our network model comprises three fundamental components as like in [15]:

- **IoT Devices:** These nodes serve as the cornerstone of the IoT network, responsible for data collection from the environment and transmitting it to the gateway nodes. This category encompasses a broad spectrum of devices, ranging from compact sensor nodes to extensive sensor-equipped industrial machinery and appliances.
- **Gateway Nodes:** Occupying a crucial role, these nodes are responsible for receiving data from the IoT devices and relaying it to the remote server, as well as the reverse. Each of these nodes also functions as a participant in the blockchain.

- **Remote Server:** This node is tasked with receiving data from the gateway nodes, processing and storing it for future use.

From an alternate perspective, our network model exhibits a two-layer structure, akin to the framework presented in [34]. The first layer is responsible for node key management and encompasses the IoT devices, while the second layer, comprising the gateway nodes, manages the blockchain aspects.

Figure 4.1 illustrates the network model of our proposed solution.

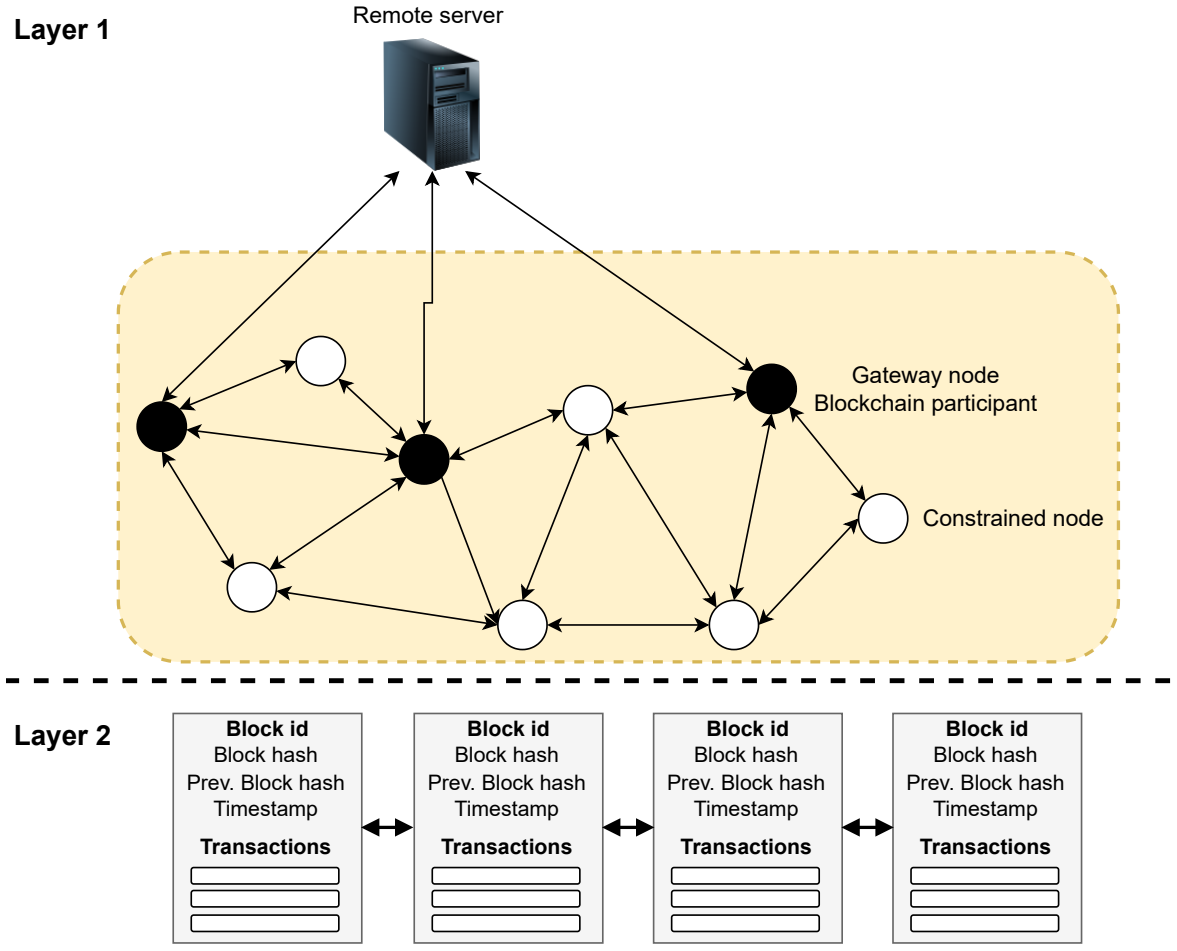


Figure 4.1: Network Model

4.3.2 Notations

Table 4.1 provides an overview of the symbols and notations employed in this chapter.

Notation	**Significance**
l_{ms}	Size of the transmitted message in bytes
l_{mr}	Size of the received message in bytes
d	Count of a node's neighboring nodes
n	Total number of nodes within the network
n_s	Quantity of member nodes within a specific set
n_{gm}	Number of member nodes associated with the gateway g
n_c	Count of cluster members

Table 4.1: Symbols and Notations

4.3.3 Assumptions

We make the following assumptions:

- The proposed solution is built upon an underlying scheme that employs blockchain technology to manage, at the very least, its key distribution phase.
- The selected consensus algorithm is PoS, chosen for its superior energy efficiency compared to PoW consensus, as documented in [53].
- The chosen blockchain type is private, it was favored for its efficiency in terms of energy consumption and transaction processing time when contrasted with public blockchains.

4.3.4 Solution steps

In this proposal, a node may be revoked under three circumstances: (1) when a node is compromised, (2) when a node voluntarily departs from the network, and (3) when a node's battery is drained.

Compromised node

If one of the blockchain participants detects that a connected constrained node has been compromised, it initiates a transaction containing the compromised node's details, including its identifier and the time of compromise. After the transaction is signed by the owner, it is broadcasted to other blockchain participants. These participants validate the transaction and append it to the blockchain by executing the associated smart contract, ensuring the node is already part of the network. Upon successful inclusion of the transaction in the blockchain, blockchain participants that have the compromised node in their list of connected nodes transmit a $KEY\_REVOCATION$ message to them. This message conveys the compromised node's identifier. Subsequently, the receiving nodes eliminate the compromised node and its key materials from their memory, responding with an ACK message to the originating blockchain participant.

Leaving node

If a node wishes to exit the network for any reason, such as transitioning to an idle mode to conserve energy, it transmits a $LEAVE$ message to its connected blockchain participant. The participant then generates a transaction encompassing the departing node's details, including its identifier, the departure request time, and the reason for leaving. This transaction is subsequently broadcasted to other blockchain participants for validation. After successful validation and addition of the transaction to the blockchain by executing the associated smart contract, which verifies the node's existing connection to the network, participants with the departing node in their connected nodes list send a $DISCONNECT$ message to their connected nodes. The $DISCONNECT$ message includes the identifier of the departing node. Receiving nodes then remove the departing node from their connected nodes list and respond with an ACK message. When the departing node receives the $DISCONNECT$ message, it erases all its keys except the pairwise key with the blockchain participant before disconnecting from the network.

Drained node

When a node's battery is drained, there is no need to transmit a $LEAVE$ message to inform other nodes of its departure. Instead, blockchain participants trigger the drain's smart contract, checking the timestamp of the last transaction to determine if the duration since the node's last interaction with the network is less than a predefined threshold. If the duration is below the threshold, a transaction is created containing the drained node's information, including its identifier and the time of depletion. Subsequently, each blockchain participant having the drained node in its list of associated nodes dispatches a $KEY\_REVOCATION$ message to its connected nodes. The $KEY\_REVOCATION$ message contains the identifier of the drained node. Receiving nodes then remove the drained node and its key materials from their memory and acknowledge the sending blockchain participant with an ACK message.

4.4 Evaluation

4.4.1 Security analysis

In this part, we conduct a comprehensive analysis of the security facets embedded within this proposal.

The security of this proposed solution is primarily underpinned by the security features of the foundational scheme upon which it is built. This scheme manages the encryption of messages exchanged between various entities within the network.

Furthermore, the transparency afforded by the blockchain architecture facilitates the auditing of the revocation process, while smart contracts enable automation. Notably, our proposed solution enhances security by refraining from storing key materials in the blockchain.

4.4.2 Performance analysis

This part entails the performance evaluation of our proposal. Initially, we provide a comparative analysis of our solution against existing solutions. Subsequently, we delve into the performance assessment of the solution, focusing on communication overhead and the energy consumption of constrained nodes.

Comparison with Existing Solutions

Solution	**Node type**	**Communication**	**Energy consumption**	**nodes count**	**Total energy consumption**
[43]	CH	$l_{ms} + (n_c - 1) * l_{mr}$	$EPSB * l_{ms} + EPRB * (n_c - 1) l_{mr}$	1	$EPSB * l_{ms} + EPRB * (n_c - 1) l_{mr}$
	CM	$l_{ms} + l_{mr}$	$EPSB * l_{ms} + EPRB * l_{mr}$	n_c	$n_c * (EPSB * l_{ms} + EPRB * l_{mr})$
[45, 47]	Gateway	$l_{ms} + n * l_{mr}$	$EPSB * l_{ms} + n * EPRB * l_{mr}$	1	$EPSB * l_{ms} + n * EPRB * l_{mr}$
	Constrained	$l_{ms} + l_{mr}$	$EPSB * l_{ms} + EPRB * l_{mr}$	n	$n * (EPSB * l_{ms} + EPRB * l_{mr})$
BKRSC-IoT	Gateway	$l_{ms} + n * l_{mr}$	$EPSB * l_{ms} + n * EPRB * l_{mr}$	1	$EPSB * l_{ms} + n * EPRB * l_{mr}$
	Constrained	$l_{ms} + l_{mr}$	$EPSB * l_{ms} + EPRB * l_{mr}$	n_{gm}	$n_{gm} * (EPSB * l_{ms} + EPRB * l_{mr})$

Table 4.2: Comparative study in case of compromised node

Table 4.2, Table 4.3, and Table 4.4 present a comparative analysis of our proposal alongside selected existing solutions in scenarios involving compromised, departing, and depleted nodes, respectively. This comparison is founded on factors such as the quantity of exchanged messages, individual node energy consumption, the number of nodes participating in the communication, and the total energy consumption across the entire network.

The process begins with an evaluation of communication overhead, achieved by summing the product of each node's transmitted and received message lengths and the number of messages exchanged. Subsequently, each node's energy consumption is determined by multiplying the calculated communication overhead by Energy per one sent byte (EPSB) and Energy per one received byte (EPRB). Finally, the total

Solution	**Node type**	**Communication**	**Energy consumption**	**nodes count**	**Total energy consumption**
[45, 47]	Leaving	$l_{ms} + d * l_{mr}$	$EPSB * l_{ms} + EPRB * d * l_{mr}$	1	$EPSB * l_{ms} + EPRB * d * l_{mr}$
	Others	$l_{ms} + lmr$	$EPSB * l_{ms} + EPRB * l_{mr}$	d	$d * (EPSB * l_{ms} + EPRB * l_{mr})$
[34]	Leaving	$(n_s - 1)l_{ms} + (n - 1)l_{mr}$	$(n_s - 1)l_{ms} * EPSB + (n - 1)l_{mr} * EPRB$	1	$(n_s - 1)l_{ms} * EPSB + (n - 1)l_{mr} * EPRB$
	Others	$l_{ms} + lmr$	$EPSB * l_{ms} + EPRB * l_{mr}$	n	$n * (EPSB * l_{ms} + EPRB * l_{mr})$
BKRSC-IoT	Gateway	$l_{ms} + (d+1)l_{mr}$	$EPSB * l_{ms} + EPRB * (d+1)l_{mr}$	1	$EPSB * l_{ms} + EPRB * (d+1)l_{mr}$
	Others	$l_{ms} + l_{mr}$	$EPSB * l_{ms} + EPRB * l_{mr}$	n_{gm}	$n_{gm} * (EPSB * l_{ms} + EPRB * l_{mr})$

Table 4.3: Comparative study in case of leaving node.

Solution	**Node type**	**Communication**	**Energy consumption**	**nodes count**	**Total energy consumption**
[45, 47]	Gateway	$l_{ms} + (n-1)l_{mr}$	$EPSB * l_{ms} + EPRB*(n-1)*l_{mr}$	1	$EPSB * l_{ms} + EPRB*(n-1)*l_{mr}$
	Constrained	$l_{ms} + (n-1)l_{mr}$	$EPSB * l_{ms} + EPRB * l_{mr}$	n	$n * (EPSB * l_{ms} + EPRB * l_{mr})$
BKRSC-IoT	Gateway	$l_{ms} + (n_{gm})l_{mr}$	$EPSB * l_{ms} + EPRB * (n_{gm})l_{mr}$	1	$EPSB * l_{ms} + EPRB * (n_{gm})l_{mr}$
	Constrained	$l_{ms} + l_{mr}$	$EPSB * l_{ms} + EPRB * l_{mr}$	n_{gm}	$n_{gm} * (EPSB * l_{ms} + EPRB * l_{mr})$

Table 4.4: Comparative study in case of drained node.

energy consumption across the network is computed by multiplying the derived node energy consumption by the number of participating nodes.

Our findings prominently prove that, when it comes to the quantity of messages exchanged and energy utilization, our solution outperforms existing alternatives. Additionally, the number of nodes engaged in the communication process is also notably reduced in our solution in contrast to the existing alternatives.

To illustrate these comparisons, let's consider an example. In the first scenario of

node compromise, the number of exchanged messages across all compared schemes, including ours, remains constant. The distinguishing factor lies in the number of involved nodes, which depends on the number of gateway member nodes and cluster members in BKRSC-IoT and [43], respectively. In the worst-case scenario, the number of involved nodes in BKRSC-IoT is less than or equal to the total number of nodes in the network compared to [43].

The second scenario, involving a departing node, similarly maintains an identical number of exchanged messages across all compared schemes, including ours. However, the number of involved nodes varies, contingent on the number of neighbors of the leaving node in [45, 47], and the number of gateway member nodes in our solution. From an alternative perspective, the number of involved nodes in [34] equals the total number of nodes in the network, representing the worst-case scenario, which makes our solution outperform [34].

In the third scenario, wherein a node's energy is depleted, the number of exchanged messages for constrained nodes remains consistent across all compared schemes, including ours. In contrast, our scheme exhibits a number of exchanged messages less than or equal to other solutions' message count in the worst-case scenario involving one gateway.

Additionally, the number of involved nodes in our scheme is also fewer than or equal to the number of involved nodes in other solutions, demonstrating the superior efficiency of our scheme.

Experimental Results (Simulation)

This section presents the simulation results of our proposed solution. The simulation was implemented using the Rust programming language, where we used it to compare our proposal with two others: [45, 47].

The simulation primarily focuses on evaluating the total communication overhead and energy consumption of nodes within the network. The setup involved deploying 100 nodes randomly across a square area of $100m^2$. The proportion of gateway member nodes was set at 10% of the total number of nodes. Each node's number of neighbors ranged from 10 to 15.

For each of the three scenarios involving compromised, departing, and drained nodes, the simulation was executed 1000 times. The simulation adhered to the model outlined in [65], which factors in an EPSB value of $59.2\mu J$ and an EPRB value of $28.6\mu J$ for the transmission and reception of one byte, respectively.

Communication overhead Figure 4.2, Figure 4.3, and Figure 4.4 depict the communication overhead necessary for all constrained nodes within the network in scenarios involving compromised, departing, and depleted nodes, respectively.

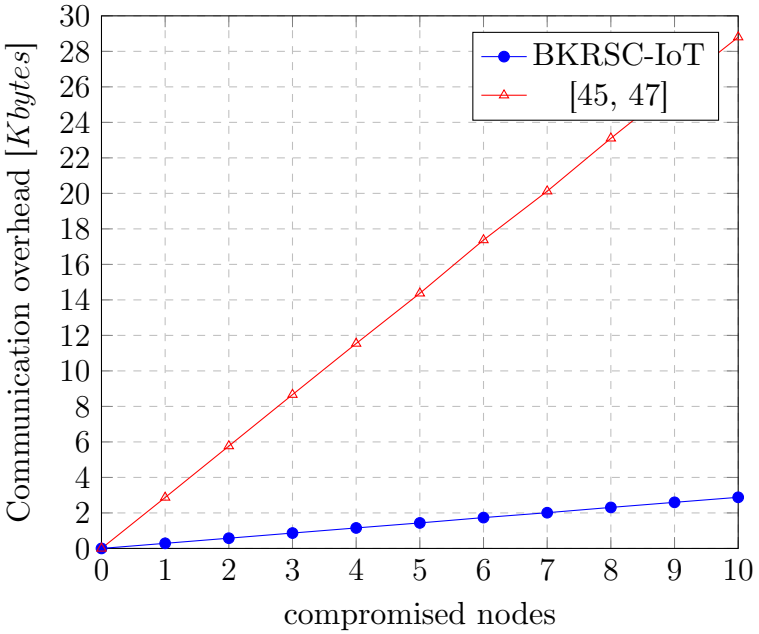


Figure 4.2: Communication overhead required by all the constrained nodes in the network in the case of compromised nodes.

Our solution exhibits greater efficiency in terms of communication overhead, as demonstrated by the results. This efficiency is attributed to the decentralization of Blockchain and the delegation of a significant portion of the communication overhead to the gateway nodes. Each gateway is tasked with communicating exclusively with its attached constrained nodes. Consequently, unlike the cases observed in [45] and [47], the number of involved nodes increases in those approaches. Additionally, in most scenarios, the bulk of communication tasks are carried out by the constrained nodes, while the gateway maintains communication with all network nodes, resulting in increased communication overhead for the constrained devices.

For the departing node scenario in our proposal, the departing node communicates

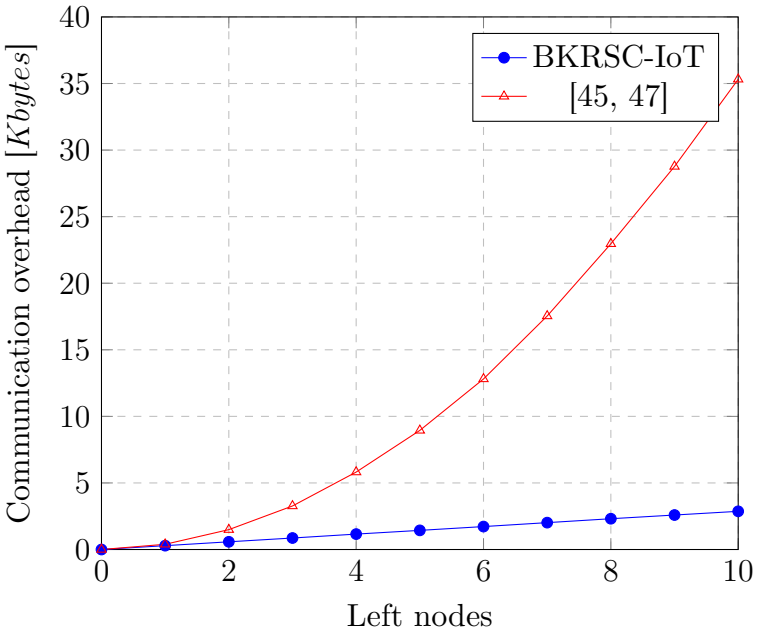


Figure 4.3: Communication overhead required by all the constrained nodes in the network in the case of left nodes.

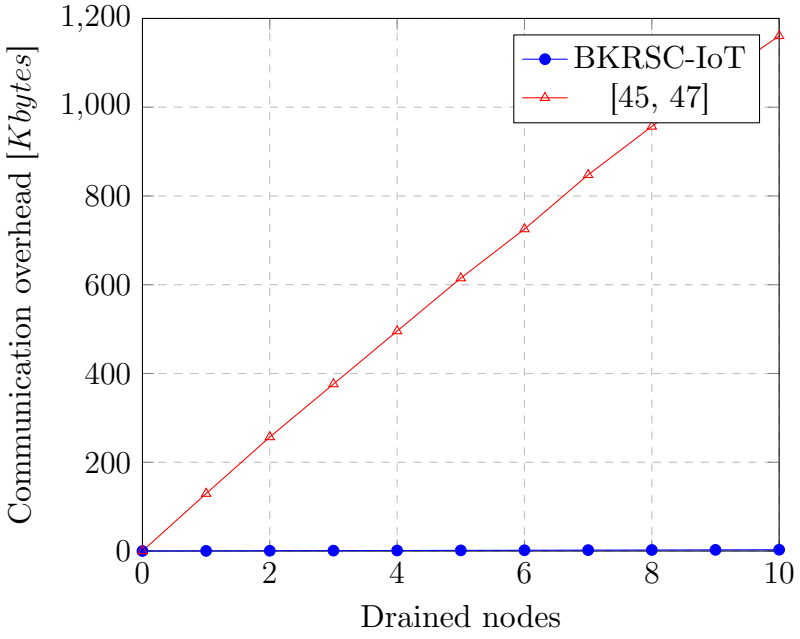


Figure 4.4: Communication overhead required by all the constrained nodes in the network in the case of drained nodes.

directly with its gateway, which, in turn, communicates with other nodes to inform them. This implies the exchange of only one message for sending and receiving. In contrast, in [45], the leaving node is required to send and receive d ACK messages,

leading to an escalation in communication overhead.

Energy Consumption :

Figure 4.5, Figure 4.6, and Figure 4.7 present the simulation results of the energy consumed by the constrained nodes as a consequence of their communications in the three scenarios involving compromised, departing, and depleted nodes, respectively.

The number of affected nodes ranges from 1 to 10. These figures illustrate that our solution surpasses existing alternatives in terms of energy efficiency for the constrained nodes in all three scenarios. This superiority is attributed to the reduced number of constrained nodes involved in communication tasks and the effective delegation of communication responsibilities to the gateway nodes.

It's noteworthy that Figure 4.5, Figure 4.6, and Figure 4.7 exhibit striking similarities to Figure 4.2, Figure 4.3, and Figure 4.4. This resemblance is attributable to the direct correlation between communication energy consumption and the number and size of messages exchanged.

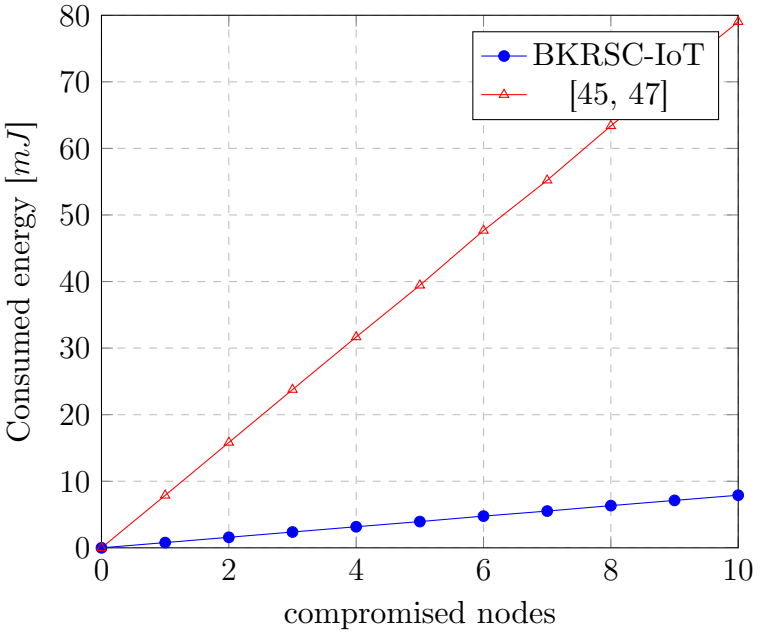


Figure 4.5: Energy consumption of communication between all the constrained nodes in the network in the case of compromised nodes.

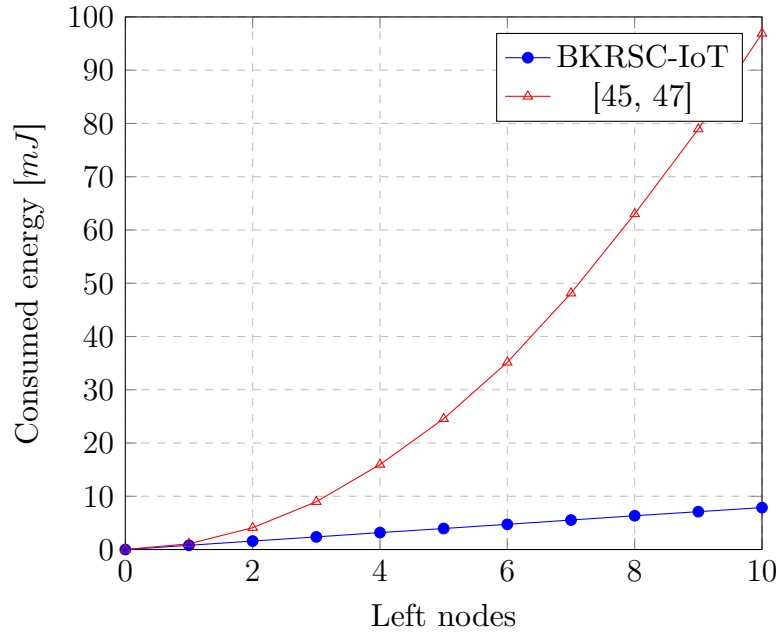


Figure 4.6: Energy consumption of communication between all the constrained nodes in the network in the case of left nodes.

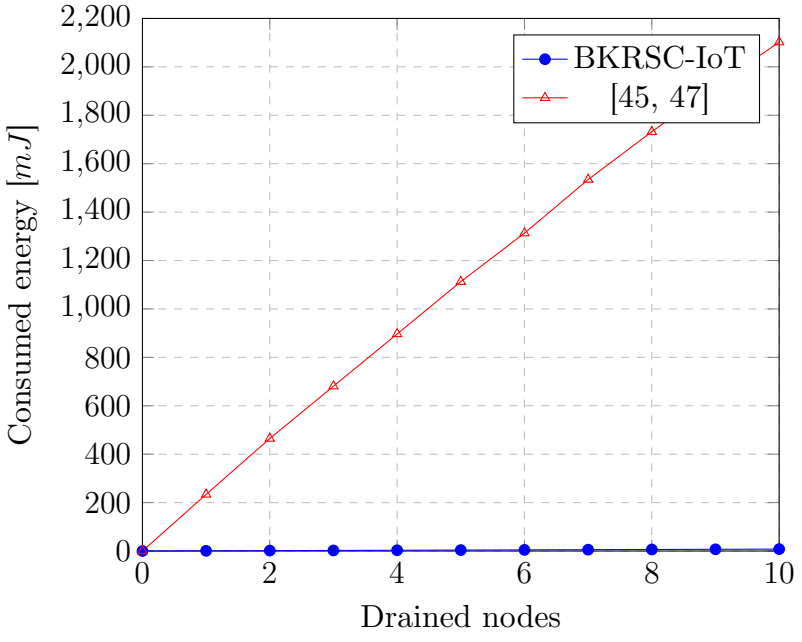


Figure 4.7: Energy consumption of communication between all the constrained nodes in the network in the case of drained nodes.

4.5 Conclusion

In conclusion, the concept of key revocation holds a pivotal position within the realm of comprehensive key management solutions. This chapter introduced a novel blockchain-based approach, underpinned by smart contracts, designed to effectively address the intricate challenge of key revocation within IoT networks.

Our proposal leverages the transparency and automation inherent in blockchain technology to establish a decentralized scheme that significantly reduces communication overhead and energy consumption. Moreover, our meticulous security analysis and rigorous performance evaluation have underscored the robust security attributes of our proposed solution, intricately linked to the security of the underlying base scheme, while notably refraining from storing any key materials within the blockchain.

Through comprehensive simulations, we have demonstrated the superior efficiency of our proposal in comparison to existing solutions. However, it is imperative to acknowledge the intrinsic limitations of this study, particularly in light of potential vulnerabilities that may arise in the event of a compromised blockchain participant.

Looking forward, our research trajectory envisions the expansion of our proposed solution to address a broader spectrum of key management challenges within IoT networks. This includes the pivotal phases of key distribution and the seamless incorporation of new nodes, promising avenues that offer potential enhancements in the security and efficiency of IoT networks.

As we conclude this chapter, the forthcoming general conclusion of this thesis will consolidate these findings and chart the course for future research directions within this dynamic and evolving field.

Conclusion and future perspectives

In conclusion, this doctoral thesis delves into the intricate realm of the IoT and its evolving significance across numerous domains. IoT's transformative potential is undeniable, as it integrates billions of smart and autonomous devices, allowing them to collect, process, and share data to provide advanced services. However, this digital revolution is accompanied by formidable security challenges, primarily because of the resource limitations of IoT devices and the inherent WSN vulnerabilities.

A critical observation within this thesis highlights the inadequacy of the traditional proposed security schemes, for safeguarding IoT networks. These protocols improve one aspect of security or performance over another, and they often fail to achieve a balance between them. This imbalance is particularly evident in key management schemes.

The core contributions of this research are embodied in two distinctive security mechanisms: EVKMS and BKRSC-IoT. EVKMS introduces a novel approach to key management, emphasizing a balanced strategy for ensuring secure communications within IoT networks. It leverages pre-distributed vectors to obfuscate stored keys and sensitive data sent via unsecured channels. Notably, EVKMS shows not only scalability and minimal computational overhead but also the straightforward integration of new nodes, all without unnecessary operations. Performance evaluations underscore EVKMS's superiority over existing schemes, making it a promising solution for IoT network security. Nevertheless, it is essential to recognize that simulations, while informative, have limitations in assessing the precise computational overhead of op-

erations employed in the scheme. Future work will involve real-world experiments to address this gap.

The second major contribution, BKRSC-IoT, focuses on the critical issue of key revocation in IoT networks. It is recognized that securing sensitive data exchange among IoT devices necessitates robust key management solutions, especially in the context of key revocation. Existing solutions require enhancement to accommodate the resource constraints of IoT devices and to reduce communication overhead. The proposed solution employs blockchain and smart contracts to minimize communication overhead and energy consumption in IoT networks, addressing these limitations. Security and performance analyses validate the effectiveness of this solution, demonstrating significant reductions in communication overhead during compromising, leaving, and draining scenarios, marking it as a robust and efficient option for IoT networks.

In terms of future perspectives, several crucial areas beckon for exploration and development. It is imperative to continue improving the efficiency of the proposed security mechanisms while sustaining their rigorous security standards. Further research should also explore the use of machine learning algorithms and blockchain technology for anomaly detection in IoT environments. These areas represent exciting avenues for further investigation, contributing to the evolving landscape of IoT security.

In an academic context, this thesis advances our understanding of IoT security and paves the way for robust, efficient, and scalable solutions to protect IoT networks in the face of evolving threats. Future research will undoubtedly build upon these foundations, seeking to fortify IoT security measures and accommodate the dynamic nature of this field.

Bibliography

[1] Orieb AbuAlghanam, Mohammad Qatawneh, Wesam Almobaideen, and Maha Saadeh. A new hierarchical architecture and protocol for key distribution in the context of iot-based smart cities. *Journal of Information Security and Applications*, 67:103173, 2022.

[2] Ian F Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: a survey. *Computer networks*, 47(4):445–487, 2005.

[3] Mishall Hammed Awaad Al-Zubaidie. *Incorporating security into electronic health records based healthcare systems with wireless sensor networks*. PhD thesis, University of Southern Queensland, 2020.

[4] Mahdi R Alagheband and Mohammad Reza Aref. Dynamic and secure key management model for hierarchical heterogeneous sensor networks. *IET Information Security*, 6(4):271–280, 2012.

[5] LoRa Alliance. Lorawan™ 1.1 specification. *LoRa Alliance*, 11:2018–04, 2017.

[6] ZigBee Alliance. Zigbee alliance. *WPAN industry group, http://www. zigbee. org/. The industry group responsible for the ZigBee standard and certification*, page 28, 2010.

[7] Ricardo S Alonso, Inés Sittón-Candanedo, Óscar García, Javier Prieto, and Sara Rodríguez-González. An intelligent edge-iot platform for monitoring livestock and crops in a dairy farming scenario. *Ad Hoc Networks*, 98:102047, 2020.

[8] Will Arthur, David Challener, and Kenneth Goldman. *A practical guide to TPM 2.0: Using the new trusted platform module in the new age of security*. Springer Nature, 2015.

[9] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[10] J Ayuso, L Marin, A Jara, and AFG Skarmeta. Optimization of public key cryptography for 16-bits devices based on 6lowpan. In *1st In. Workshop on Security of the Internet of Things, Japan*, pages 1–8, 2010.

[11] E Baburaj et al. Polynomial and multivariate mapping-based triple-key approach for secure key distribution in wireless sensor networks. *Computers & Electrical Engineering*, 59:274–290, 2017.

[12] Linyan Bai, Chingfang Hsu, Lein Harn, Jianqun Cui, and Zhuo Zhao. A practical lightweight anonymous authentication and key establishment scheme for resource-asymmetric smart environments. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[13] Sami Bettayeb. samibettayeb/evkms-iot-simulation: 2.0.0, May 2023.

[14] Sami Bettayeb. samibettayeb/resilience-simulation: 1.0.0, April 2023.

[15] Sami Bettayeb, Mohamed-Lamine Messai, and Sofiane Mounine Hemam. A robust and efficient vector-based key management scheme for iot networks. *Ad Hoc Networks*, 149:103250, 2023.

[16] Rolf Blom. An optimal class of symmetric key generation systems. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 335–338. Springer, 1984.

[17] SIG Bluetooth. Bluetooth core specification version 4.0. *Specification of the Bluetooth System*, 1(7):206, 2010.

[18] Martin Bor, John Edward Vidler, and Utz Roedig. Lora for the internet of things. 2016.

[19] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems (TOCS)*, 8(1):18–36, 1990.

[20] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[21] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *2003 Symposium on Security and Privacy, 2003.*, pages 197–213. IEEE, 2003.

[22] Wenliang Du, Jing Deng, Yunghsiang S Han, Pramod K Varshney, Jonathan Katz, and Aram Khalili. A pairwise key predistribution scheme for wireless

sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, 8(2):228–258, 2005.

[23] Morris Dworkin, Elaine Barker, James Nechvatal, James Foti, Lawrence Bassham, E. Roback, and James Dray. Advanced encryption standard (aes), 2001-11-26 2001.

[24] Mohamed Eltoweissy, Mohammed Moharrum, and Ravi Mukkamala. Dynamic key management in sensor networks. *IEEE Communications magazine*, 44(4):122–130, 2006.

[25] Christian Esposito, Massimo Ficco, Aniello Castiglione, Francesco Palmieri, and Alfredo De Santis. Distributed group key management for event notification confidentiality among sensors. *IEEE Transactions on Dependable and Secure Computing*, 17(3):566–580, 2018.

[26] Kai Fan, Hui Li, and Yue Wang. Security analysis of the kerberos protocol using ban logic. In *2009 Fifth International Conference on Information Assurance and Security*, volume 2, pages 467–470. IEEE, 2009.

[27] Amit Kumar Gautam and Rakesh Kumar. A key management scheme using (p, q)-lucas polynomials in wireless sensor network. *China Communications*, 18(11):210–228, 2021.

[28] IEEE Working Group et al. Wireless medium access control and physical layer specifications for low-rate wireless personal area networks. *IEEE Standard*, 802(4):2003, 2003.

[29] ITU-T Study Group et al. New itu standards define the internet of things and provide the blueprints for its development, 2012.

[30] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[31] Utku Gulen and Selcuk Baktir. Elliptic curve cryptography for wireless sensor networks using the number theoretic transform. *Sensors*, 20(5):1507, 2020.

[32] Saleem S Hameedi and Oguz Bayat. Improving iot data security and integrity using lightweight blockchain dynamic table. *Applied Sciences*, 12(18):9377, 2022.

[33] Christian Huitema. *IPv6: the new Internet protocol*. Prentice-Hall, Inc., 1995.

[34] Mohamed Ali Kandi, Djamel Eddine Kouicem, Messaoud Doudou, Hicham Lakhlef, Abdelmadjid Bouabdallah, and Yacine Challal. A decentralized blockchain-based key management protocol for heterogeneous and dynamic iot devices. *Computer Communications*, 191:11–25, 2022.

[35] Luke E Kane, Jiaming James Chen, Rebecca Thomas, Vicky Liu, and Matthew Mckague. Security and performance in iot: A balancing act. *IEEE access*, 8:121969–121986, 2020.

[36] Steven L Kinney. *Trusted platform module basics: using TPM in embedded systems*. Elsevier, 2006.

[37] Daniel Larimer. Delegated proof-of-stake (dpos). *Bitshare whitepaper*, 81:85, 2014.

[38] Ao Lei, Haitham Cruickshank, Yue Cao, Philip Asuquo, Chibueze P Anyigor Ogah, and Zhili Sun. Blockchain-based dynamic key management for heterogeneous intelligent transportation systems. *IEEE Internet of Things Journal*, 4(6):1832–1843, 2017.

[39] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE internet of things journal*, 4(5):1125–1142, 2017.

[40] Zhe Liu, Xinyi Huang, Zhi Hu, Muhammad Khurram Khan, Hwajeong Seo, and Lu Zhou. On emerging family of elliptic curves to secure internet of things: Ecc comes of age. *IEEE Transactions on Dependable and Secure Computing*, 14(3):237–248, 2016.

[41] Mingxin Ma, Guozhen Shi, and Fenghua Li. Privacy-oriented blockchain-based distributed key management architecture for hierarchical access control in the iot scenario. *IEEE access*, 7:34045–34059, 2019.

[42] Dieynaba Mall, Karim Konaté, and Al-Sakib Khan Pathan. Ecl-ekm: An enhanced certificateless effective key management protocol for dynamic wsn. In *2017 International conference on networking, Systems and Security (NSysS)*, pages 150–155. IEEE, 2017.

[43] Samira Mesmoudi, Belkacem Benadda, and Amin Mesmoudi. Skwn: Smart and dynamic key management scheme for wireless sensor networks. *International Journal of Communication Systems*, 32(7):e3930, 2019.

[44] Amina Msolli, Nader Ajmi, Abdelhamid Helali, Abdelaziz Gassoumi, Hassen Maaref, and Ridha Mghaieth. New key management scheme based on pool-hash for wsn and iot. *Journal of Information Security and Applications*, 73:103415, 2023.

[45] Mohammed Nafi, Samia Bouzefrane, and Mawloud Omar. Matrix-based key management scheme for iot networks. *Ad Hoc Networks*, 97:102003, 2020.

[46] Mohammed Nafi, Samia Bouzefrane, and Mawloud Omar. Efficient and lightweight polynomial-based key management scheme for dynamic networks. In *International Conference on Mobile, Secure, and Programmable Networking*, pages 110–122. Springer, 2021.

[47] Mohammed Nafi, Mohamed-Lamine Messai, Samia Bouzefrane, and Mawloud Omar. Ifkms: Inverse function-based key management scheme for iot networks. *Journal of Information Security and Applications*, 71:103370, 2022.

[48] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, page 21260, 2008.

[49] Levent Özgür, Vahid Khalilpour Akram, Moharram Challenger, and Orhan Dağdeviren. An iot based smart thermostat. In *2018 5th International Conference on Electrical and Electronic Engineering (ICEEE)*, pages 252–256. IEEE, 2018.

[50] Ruchi Parashar, A Khan, and AK Neha. A survey: The internet of things. *International Journal of Technical Research and Applications*, 4(3):251–257, 2016.

[51] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials*, 16(1):414–454, 2013.

[52] S Poovarasan and R Brindha. Survey of various techniques to reduce leakage power in fpga. *International Journal of communication and computer Technologies*, 4(1):23–28, 2016.

[53] Fahad Saleh. Blockchain without waste: Proof-of-stake. *The Review of financial studies*, 34(3):1156–1190, 2021.

[54] Zach Shelby, Klaus Hartke, and Carsten Bormann. The constrained application protocol (coap). Technical report, 2014.

[55] Zhengguo Sheng, Chinmaya Mahapatra, Chunsheng Zhu, and Victor CM Leung. Recent advances in industrial wireless sensor networks toward efficient management in iot. *IEEE access*, 3:622–637, 2015.

[56] Soram Ranbir Singh, Ajoy Kumar Khan, and Takhellambam Sonamani Singh. A new key management scheme for wireless sensor networks using an elliptic curve. *Indian Journal of Science and Technology*, 10(13):1–7, 2017.

[57] Nick Szabo. Formalizing and securing relationships on public networks. *First monday*, 1997.

[58] Marco Tiloca and Gianluca Dini. Grep: A group rekeying protocol based on member join history. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 326–333. IEEE, 2016.

[59] Tanya Mohan Tukade and R Banakar. Data transfer protocols in iot—an overview. *Int. J. Pure Appl. Math*, 118(16):121–138, 2018.

[60] Pavel Vasin. Blackcoin's proof-of-stake protocol v2. *URL: https://blackcoin.org/blackcoin-pos-protocol-v2-whitepaper.pdf*, 71, 2014.

[61] Luca Veltri, Simone Cirani, Stefano Busanelli, and Gianluigi Ferrari. A novel batch-based group key management protocol applied to the internet of things. *Ad Hoc Networks*, 11(8):2724–2737, 2013.

[62] Ovidiu Vermesan and Peter Friess. *Internet of things: converging technologies for smart environments and integrated ecosystems*. River publishers, 2013.

[63] P Vijayakumar, S Bose, and A Kannan. Rotation based secure multicast key management for batch rekeying operations. *Networking Science*, 1:39–47, 2012.

[64] Wattana Viriyasitavat and Danupol Hoonsopon. Blockchain characteristics and consensus in modern business processes. *Journal of Industrial Information Integration*, 13:32–39, 2019.

[65] Arvinderpal S Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Third IEEE international conference on pervasive computing and communications*, pages 324–328. IEEE, 2005.

[66] Qian Wang, Hai Su, Kui Ren, and Kwangjo Kim. Fast and scalable secret key generation exploiting channel phase randomness in wireless networks. In *2011 Proceedings IEEE INFOCOM*, pages 1422–1430. IEEE, 2011.

[67] Zhen Yu and Yong Guan. A key management scheme using deployment knowledge for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(10):1411–1425, 2008.

[68] Aiqing Zhang and Xiaodong Lin. Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain. *Journal of medical systems*, 42(8):140, 2018.

[69] Yuexin Zhang, Yang Xiang, Xinyi Huang, Xiaofeng Chen, and Abdulhameed Alelaiwi. A matrix-based cross-layer key establishment protocol for smart homes. *Information Sciences*, 429:390–405, 2018.

List of publications

Journal Publications

- Sami Bettayeb, Mohamed-Lamine Messai, Sofiane Mounine Hemam, "A robust and efficient vector-based key management scheme for IoT networks," *Ad Hoc Networks*, 149, 2023, Elsevier.

Conference Proceedings

- Sami Bettayeb, Mohamed-Lamine Messai, Sofiane Mounine Hemam, "BKRSCT-IoT: Blockchain-based Key Revocation using Smart Contracts for IoT networks," *International Conference on Management of Digital Ecosystems*, pages 331–344. Springer, 2023.