The Indonesian Journal of Computer Science
www.ijcs.net
Volume 14, Issue 5, October 2025
https://doi.org/10.33022/ijcs.v14i5.4986

UML Modeling of Digital Envelope-Based Security System to Fulfill Security Services

Isa Mulia Insan[1], Jati Hiliamsyah Husen[2], Rahmat Yasirandi[3]
isamuliainsan@telkomuniversity.ac.id[1], jatihusen@telkomuniversity.ac.id[2],
batanganhitam@telkomuniversity.ac.id[3]
[1,2]School of Computing, Telkom University, Bandung, Indonesia
[3]CoE Technological Society (CAATIS), Telkom University, Bandung, Indonesia

Article Information

Received : 27 Aug 2025
Revised : 7 Oct 2025
Accepted : 23 Oct 2025

Keywords

UML Modeling, Digital Envelope, Encryption Systems, Security Services, IoT Security

Abstract

This research focuses on the development and systematic redesign of a Unified Modeling Language (UML) model to represent a digital envelope-based security system. The digital envelope method, integrating both symmetric and asymmetric encryption, is employed to leverage the strengths of each encryption type, ensuring performance and security in data protection. The study enhances a previously proposed UML model by incorporating security symbols and notations tailored for security modeling, effectively capturing encryption principles while ensuring clarity and accuracy. However, several limitations were identified, particularly the lack of detailed separation between the encryption and decryption processes, which are crucial for ensuring data integrity, confidentiality, and non-repudiation. The research concludes that further development is required to refine these notations, including a clear distinction between encryption and decryption stages, and the inclusion of more detailed symbols for key management. Future work should focus on extending the notations to better address the security challenges faced by digital envelope-based systems, enhancing their representation of key generation, storage, and distribution.

A. Introduction

Technology has been widely adopted across various sectors such as government, industry, and academia[1]. As the adoption of technology increases, the number of requirements that must be fulfilled also expands. The ITU's X.800 document outlines five essential categories of security services that must be provided: ensuring the identity of the involved parties (authentication), managing access rights to data (access control), maintaining the confidentiality of the message content (data confidentiality), ensuring that the data remains unaltered (data integrity), and preventing the denial of actions taken (non-repudiation)[2]. These services are essential to ensure the security of a system, particularly information systems implementing the Internet of Things (IoT) concept.

The primary challenge in the implementation of IoT within networks is preventing unauthorized individuals from gaining access to transmitted data[3]. In healthcare settings, IoT is utilized for the continuous monitoring of patients' vital signs[4]. Vital data such as blood pressure, heart rate, and blood oxygen levels are monitored and transmitted to healthcare providers or other authorized personnel. Unauthorized modifications to this data could lead to harmful medical decisions. Therefore, ensuring patient security is of paramount importance in IoT applications [5]. Encryption is an effective technique for protecting data transmission in IoT systems, particularly in the healthcare sector [6]. It is an essential technique for safeguarding data by transforming the message into an unreadable format (ciphertext). The primary objective is to ensure that the message can only be accessed by authorized individuals possessing the decryption key. However, the implementation of encryption in IoT devices faces the challenge of limited resources, necessitating the adaptation of encryption techniques to accommodate these constraints.

Two main categories of encryption are recognized, distinguished by the encryption and decryption mechanisms. Symmetric encryption is well-known for its efficiency and uses a single key for both encryption and decryption. In contrast, asymmetric encryption, which relies on two distinct keys, is regarded as more secure. A hybrid approach, known as the digital envelope [7], combines symmetric and asymmetric encryption to harness the advantages of both. Four of the five security services outlined have been successfully addressed by security systems employing the digital envelope [8]. In these studies, the proposed security scheme effectively satisfies system requirements, including the ability to operate on constrained devices with limited resources.

The development of a system or software typically begins with requirements engineering to accurately identify user needs. Requirements engineering can be defined as the process of transforming user needs and desires into a concrete, implementable technical blueprint. System requirements are one level within the Requirements Level Classification scheme defined by Sommerville[9], consisting of detailed descriptions of services and constraints. These descriptions can be represented graphically to aid in explanation. It has been shown that graphical models are more effective than textual descriptions in facilitating discussions among developers[10]. UML (Unified Modeling Language) is a standard modeling language widely supported by various graphical tools and is well-understood by most developers[11]. Furthermore, in the context of IoT systems, there is currently

no established standardization for a representative language that can effectively capture the diverse requirements and complexities of these systems[12].

The objective of this study is to methodically design a UML model that accurately represents the current digital envelope system, incorporating predefined security symbols and notations [13]. The proposed model incorporates security symbols and notations, which are employed to design diagrams that enhance security requirements modeling. Furthermore, this approach enables better representation of security mechanisms within software design.

B. Related Works

Encryption is a method of securing data that has been widely utilized to protect sensitive information. Two commonly used encryption techniques are symmetric and asymmetric encryption. A method that combines both of these techniques is known as the digital envelope method.

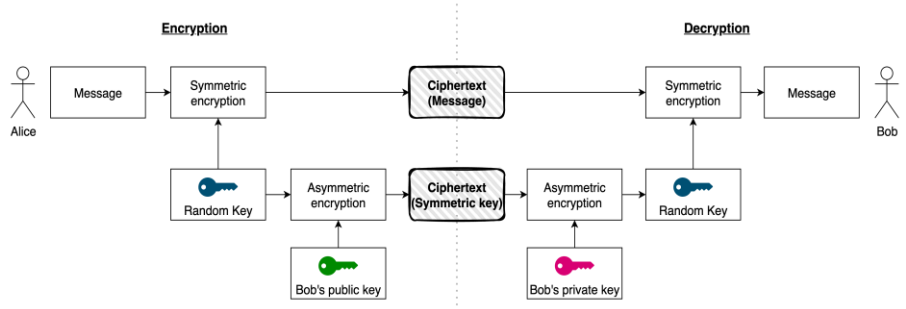

Figure 1. Digital envelope [8]

The application of this method can provide four of the five security services defined by ITU X.800[2], which are: (1) ensuring that communication between entities is legitimate (authentication), (2) controlling access to allow only authorized access (access control), (3) securing data from unauthorized disclosure (data confidentiality), and (4) ensuring that the message received or sent has not been modified or corrupted (data integrity). The fifth service, ensuring that the sender cannot deny sending the message (non-repudiation), is also critical but was not directly addressed in this context.

As quoted from the book Requirements Engineering for Software and Systems, "*Requirements engineering is the branch of engineering concerned with the real-world goals for, functions of, and constraints on systems. It is also concerned with the relationship of these factors to precise specifications of system behavior and to their evolution over time and across families of related systems*"[14]. In this context, real-world goals refer to the users or stakeholders involved; functions describe what the system must be able to perform; and constraints include technical limitations, resource restrictions, or regulatory requirements that must be considered during system development. Thus, requirements engineering can be understood as the process of transforming user needs and expectations into a technical blueprint that can be implemented.

The widespread use of UML (Unified Modeling Language) in software development has prompted research that integrates AI (ChatGPT), to design UML diagrams[15]. ChatGPT has been employed to generate sequence diagrams from natural language requirements. The reliability of ChatGPT has been evaluated based

on five criteria using a scoring scale. Other studies have implemented UML to model security systems for IoT [13], [16], [17]. All three studies agree that there is no standard approach for designing security within IoT architectures. Other studies have implemented UML to model security systems for IoT. The first study[16] proposes an architecture that can be used across four IoT layers. The study concludes that IoT modeling in real-world environments is considered difficult, complex, time-consuming, and costly. The second study[17] introduces a plug-in tool that can assist in designing IoT systems.

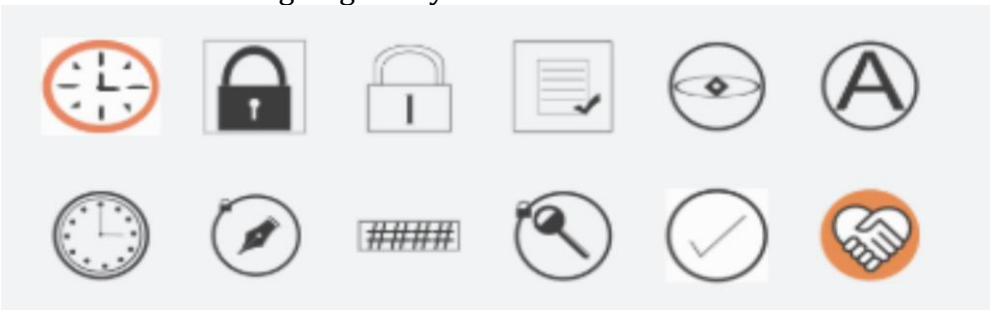

Figure 2. Security symbols and notations [13]

As quoted from the research[13] "*The author's argument regarding notations is that they lack attention to security concerns*". This study proposed a model of security symbols and notations to explain security within a graphical context. Figure 2 shows the proposed security symbols and notations that can be implemented. The symbols are arranged from top left as follows: availability, access control, integrity, data integrity, sensory experience, authorization, session, non-repudiation, hashing, encryption, authentication, and confidentiality. These symbols and notations will be used as needed to design models for system requirements.

C. Research Method

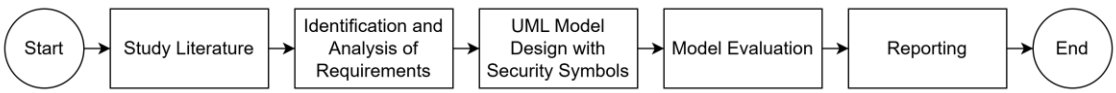

Figure 3. Research method diagram

The research flow has been illustrated in Figure 3, follows the stages of literature review, identification and analysis of requirements, UML model design with security symbols, model evaluation, and report writing. The literature review focuses on the security concepts provided by the digital envelope in accordance with the ITU X.800 definitions. It then explores software requirements engineering and the use of UML in security modeling. After conducting a literature review on both topics, the system requirements identified from previous research are: " How UML modeling using security symbols and notations can be applied to the system requirements of a digital envelope-based security system."

The next stage is the identification and analysis of requirements. Identification is carried out on several Secure UMLs that can be used in modeling. The outcome is the selection of security symbols and notations to be used in the following stage. The next step is the design of the UML model. Based on the UML designed in previous

studies, modifications are made using the new notations. The results are evaluated in the next stage with the team and practitioners to gain feedback on the model used. All stages and the results obtained are documented in the final phase for dissemination.

D. Result and Discussion

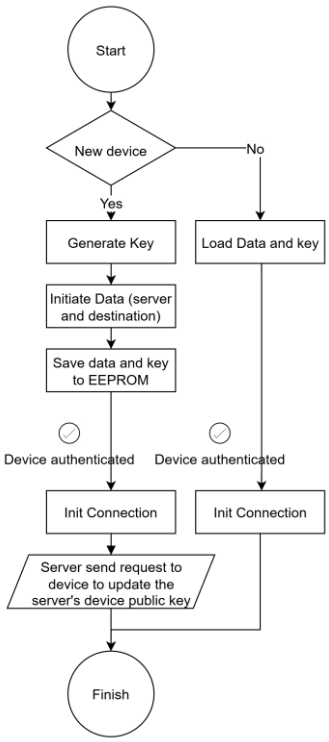

Figure 4. Initiate phase activity diagram

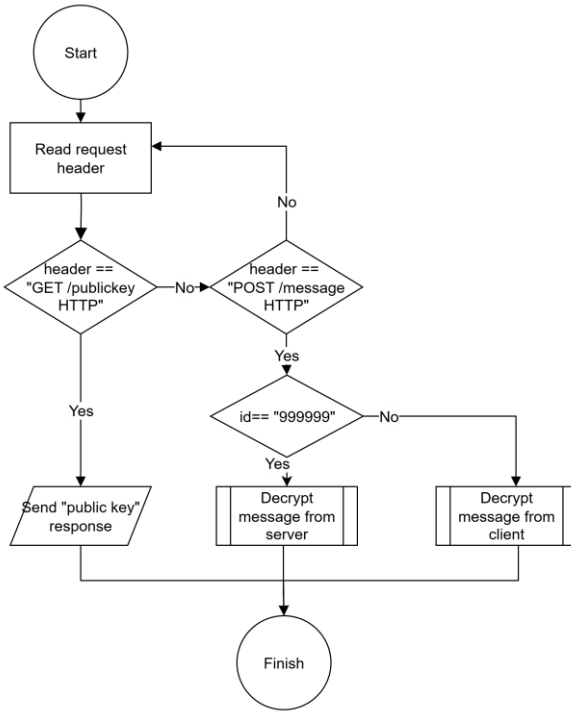

Figure 5. Checking header activity diagram

In Figure 4, the initiation phase of the proposed security system is shown, which occurs before the device can communicate. If the device has not been initialized before, it will generate a new key pair (public key and private key). These keys are used to encrypt and decrypt messages during communication. After that, the device stores the key data, server information, and destination data in the EEPROM (Electrically Erasable Programmable Read-Only Memory). Once the keys are successfully stored, the device is considered authenticated. The next step is for the server to obtain the device's public key by sending a request to the client and updating the public key data on the server. The received public key will be stored on the server, thus allowing the server to have all the public keys of the clients. If the device has been initialized previously, it only needs to retrieve the stored data from the EEPROM and is ready to proceed to the communication phase.

In Figure 5, an activity diagram for checking the header is presented. In this stage, the device is ready to receive requests from the web server or other devices. After the device receives a request, it will check the header of the request. If the header is "GET /publickey HTTP", the device will send the public key in the response, and the header check process is complete. If the header is "PUSH /message

HTTP", the device will check the ID contained in the request body. If the sender's ID is not the server's ID, the device will run the subprocess for checking the message from the client. However, if the sender's ID is the server's ID, the device will run the subprocess for checking the message from the server.

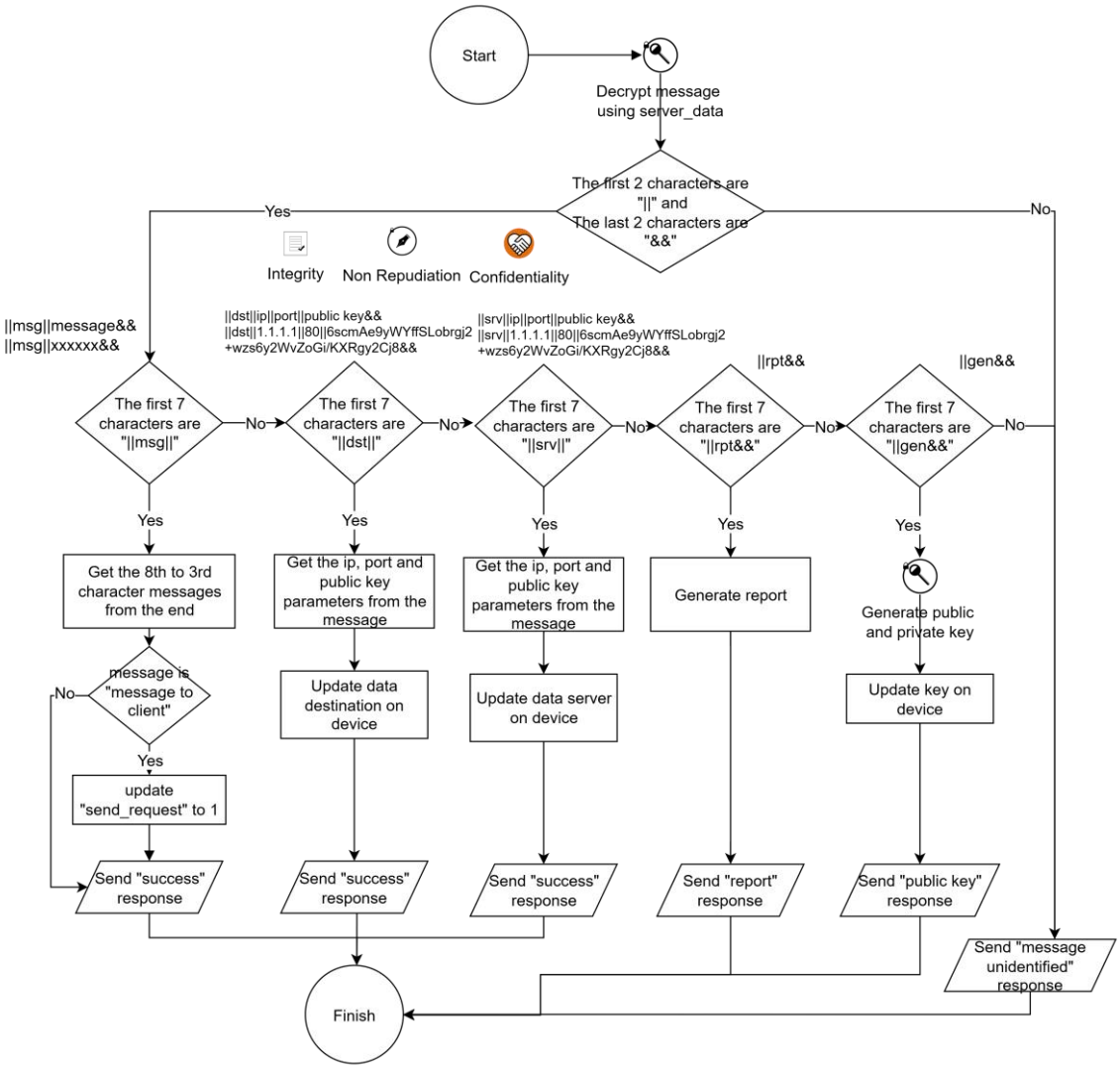

Figure 6. Checking messages from server activity diagram

The next check is for the message from the server. The activity diagram for checking the message from the server is shown in Figure 6. The body of the received request will be decrypted using the server's public key, which was stored in the server data during the initiation phase. After the message is decrypted, the format of the decrypted result will be examined. Several message formats that are accepted begin with "||msg||" followed by the message content, ending with "&&". Secondly, a message format that starts with "||dst||" includes the destination IP, destination port, and destination public key. Thirdly, a format starting with "||srv||" is similar to the second, but is used to change the device's server data. Fourthly, the message format is "||gen&&", indicating a command for the device to generate a new key.

The first check is to ensure that the first two characters are "||" and the last two characters are "&&". If this format is correct, the device will proceed with

further checks; otherwise, the device will send a response stating that the message format is not recognized. If the beginning and end of the decrypted message match, it indicates that the message's integrity, non-repudiation, and confidentiality have been maintained.

Next, the device will examine the first seven characters of the message. If the first seven characters are "||msg||", the device will extract the message content from between the eighth character and the third-to-last character. If the message contains "send message", the device will set the send_message variable to 1, indicating that the device should send the message to another device. If the message contains "report", the send_message variable will be set to 2, indicating that the device should send the message to the server.

In the second and third formats, if the first seven characters are "||dst||", it means the server wants to change the device's destination data, while if it starts with "||srv||", the server wants to change the device's server data. The data includes the destination IP, destination port, and destination public key, which will then be stored in the device's EEPROM. Afterward, the device will send a success response to the server.

The fourth format is message verification with the format "||rpt&&". A message with this format indicates that the server wants the device to send a report back to the server. The destination IP, port, and public key are server data stored in the EEPROM.

Finally, if the message format begins with "||gen&&", the device will be requested to generate a new key. The new key will be stored in the EEPROM, replacing the old key, and the new key will then be sent back as a response to update the device's public key on the server.

The activity diagram for checking messages from the client can be seen in Figure 7. The process begins with the device sending a request to the server to obtain the sender's public key. If the response received is the public key, the message will be decrypted using that key. However, if the response is not the public key, the device will immediately send a response stating that the message is unrecognized. After the message is decrypted using the server's public key, the next step is to check the message format. The format of the message must match the pattern "||msg||message content&&", where "||" is chosen because it is rarely used in sentences, and "&&" is chosen as the end delimiter because the "&" character is rarely used twice consecutively or at the end of a sentence. The check is performed by ensuring that the first seven characters are "||msg||" and the last two characters are "&&". If the message format is incorrect, the device will send a response stating that the message is unrecognized. If the message format is correct, the device will extract the message content from the eighth character to the third-to-last character.

In Figure 8, the activity diagram for sending a message from the device is shown. In the message checking section, the value of send_message can be set to "1" or "2". If send_message is "1", the device will encrypt the message using the destination data and send a request containing the encrypted message to the destination address. Afterward, send_message will be set to "0". Conversely, if send_message is "2", the device will encrypt the data using the server's data and send the encrypted message to the server address. The encrypted message is used

to maintain its integrity, non-repudiation, and confidentiality during transmission. Then, send_message will be set back to "0".

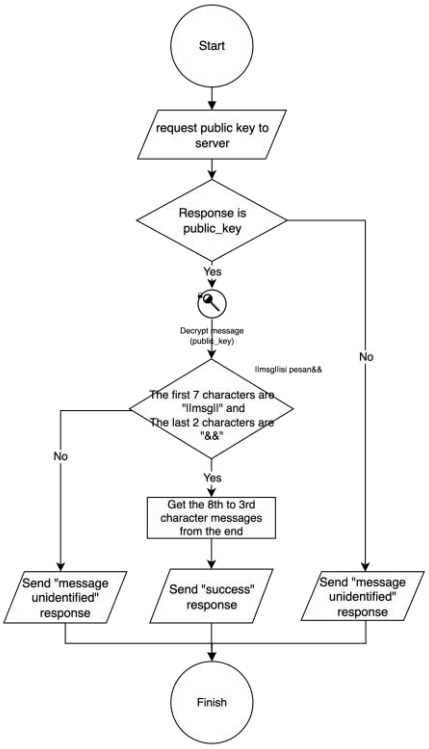

Figure 7. Activity diagram of checking messages from clients

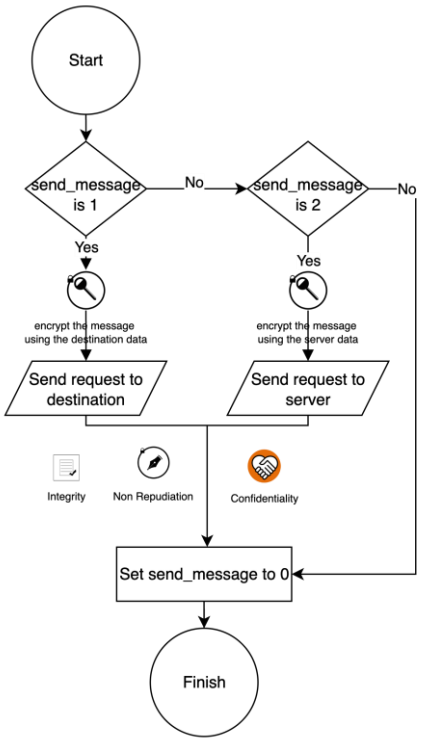

Figure 8. Sending message activity diagram

From the application of security symbols and notations for UML, several evaluations were obtained. The symbols and notations used are more suitable for explaining user requirements rather than system requirements. The article [18] discusses the difference between user and system requirements. To fulfill user requirements, the approaches that can be used are more varied than those for fulfilling system requirements. Consequently, to meet system requirements, more detailed notations are necessary. In the context of encryption systems, the security requirements are more complex and precise, meaning the model may not fully represent the system's needs. This limitation could affect the accuracy of the conclusions about how the UML model addresses the security requirements of a system.

E. Conclusion

This research has successfully developed and systematically restructured a UML model to represent a digital envelope-based security system. The digital envelope method, which combines symmetric and asymmetric encryption, capitalizes on the strengths of both encryption types. Symmetric encryption provides efficiency with a single key for both encryption and decryption, while asymmetric encryption offers enhanced security with the use of two keys (public and private). By merging these two encryption techniques, the digital envelope achieves both the performance and security necessary for effective data protection.

The UML model proposed in previous studies has been successfully modified by incorporating security symbols and notations specifically designed for UML [13].

These security notations provide a clear framework for modeling security aspects of the system, allowing for a more comprehensive understanding of how the digital envelope can be implemented in a real-world system. Despite the advancements made with these notations, several limitations remain that must be addressed in future work. One significant issue is that the current notations primarily focus on the encryption process and do not fully separate the encryption and decryption stages in sufficient detail. This is critical because, in a digital envelope-based system, the integrity and confidentiality of the data depend on how securely both the encryption and decryption processes are handled. Without a clear distinction between these stages, the model may fail to fully represent the nuanced security requirements of such systems.

Further research and development are needed to refine these notations by clearly separating the encryption and decryption processes. This will enhance the accuracy and completeness of the security modeling and ensure that all stages of the data protection process are captured in a detailed and structured manner. Additionally, the introduction of more granular symbols for key management (such as key generation, storage, and distribution) could provide a more complete security solution, addressing potential vulnerabilities in the system.

F. References

The author would like to express sincere gratitude to all parties who have provided support, assistance, and contributions in completing this research. Special thanks are also extended to colleagues, peers, and everyone who has offered valuable insights and guidance throughout the research process.

G. References

[1] R. Yasirandi, A. Lander, H. R. Sakinah, and I. M. Insan, "IoT Products Adoption for Smart Living in Indonesia: Technology Challenges and Prospects," in *2020 8th International Conference on Information and Communication Technology (ICoICT)*, IEEE, Jun. 2020, pp. 1–6. doi: 10.1109/ICoICT49345.2020.9166200.

[2] Chroust. Gerhard, Doucek. Petr, and O. Václav, *IDIMT-2022 : Digitalization of Society, Business and Management in a Pandemic : 30th Interdisciplinary Information Management Talks, Sept. 7-9, 2022, Prague, Czech Republic*. Trauner Verlag Universität, 2022.

[3] O. Panahi, "Secure IoT for Healthcare," *European Journal of Innovative Studies and Sustainability*, vol. 1, no. 1, pp. 17–23, Jan. 2025, doi: 10.59324/ejiss.2025.1(1).03.

[4] B. Alamri, K. Crowley, and I. Richardson, "Cybersecurity Risk Management Framework for Blockchain Identity Management Systems in Health IoT," *Sensors*, vol. 23, no. 1, p. 218, Dec. 2022, doi: 10.3390/s23010218.

[5] V. Vakhter, B. Soysal, P. Schaumont, and U. Guler, "Threat Modeling and Risk Analysis for Miniaturized Wireless Biomedical Devices," *IEEE Internet Things J*, vol. 9, no. 15, pp. 13338–13352, Aug. 2022, doi: 10.1109/JIOT.2022.3144130.

[6] D. Clemente-Lopez, J. de J. Rangel-Magdaleno, and J. M. Muñoz-Pacheco, "A lightweight chaos-based encryption scheme for IoT healthcare systems,"

Internet of Things, vol. 25, p. 101032, Apr. 2024, doi: 10.1016/j.iot.2023.101032.

[7] G. Uma Maheswari, A. S, C. Rajeshkumar, M. Vargheese, G. Nallasivan, and J. H. Selvarani, "Multimedia Wireless Sensor Network Platform Data Encryption Algorithm based on Blockchain Technology," in *2024 2nd International Conference on Networking and Communications (ICNWC)*, IEEE, Apr. 2024, pp. 1–7. doi: 10.1109/ICNWC60771.2024.10537414.

[8] I. M. Insan and F. Samopa, "Implementation of Http Security Protocol for Internet of Things Based on Digital Envelope," *Procedia Comput Sci*, vol. 234, pp. 1332–1339, 2024, doi: 10.1016/j.procs.2024.03.131.

[9] Ian. Sommerville, *Software engineering : seventh edition Ian Sommerville*. 2004.

[10] R. Jolak *et al.*, "Software engineering whispers: The effect of textual vs. graphical software design descriptions on software design communication," *Empir Softw Eng*, vol. 25, no. 6, pp. 4427–4471, Nov. 2020, doi: 10.1007/s10664-020-09835-6.

[11] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML: A UML-Based Modeling Language for Model-Driven Security."

[12] M. Hind, O. Noura, M. Sanae, and A. Abraham, "A Comparative Study for Modeling IoT Security Systems," 2023, pp. 258–269. doi: 10.1007/978-3-031-35510-3_25.

[13] S. M. J. Hassan, A. Shahab, F. A. Tabba, M. Alrammal, F. Abu-Amara, and M. Nadeem, "A comparison of approaches for modeling software security requirements using unified modeling language extensions," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 15, no. 3, p. 2911, Jun. 2025, doi: 10.11591/ijece.v15i3.pp2911-2927.

[14] P. A. Laplante and M. H. Kassab, *Requirements Engineering for Software and Systems*. New York: Auerbach Publications, 2022. doi: 10.1201/9781003129509.

[15] A. Ferrari, S. Abualhaija, and C. Arora, "Model Generation with LLMs: From Requirements to UML Sequence Diagrams," in *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*, IEEE, Jun. 2024, pp. 291–300. doi: 10.1109/REW61692.2024.00044.

[16] H. Meziane and N. Ouerdi, "A Study of Modelling IoT Security Systems with Unified Modelling Language (UML)," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 11, 2022, doi: 10.14569/IJACSA.2022.0131130.

[17] H. Cardenas, R. Zimmerman, A. R. Viesca, M. Al Lail, and A. J. Perez, "Formal UML-based Modeling and Analysis for Securing Location-based IoT Applications," in *2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, IEEE, Oct. 2022, pp. 722–723. doi: 10.1109/MASS56207.2022.00109.

[18] N. Maiden, "User Requirements and System Requirements," *IEEE Softw*, vol. 25, no. 2, pp. 90–91, Mar. 2008, doi: 10.1109/MS.2008.54.