

Zero-Trust Communication between Chips

Kais Belwafi
C2PS center,
Khalifa University,
Abu Dhabi, UAE
kais.belwafi@ku.ac.ae

Hamdan Alshamsi
EECS department,
Khalifa University,
Abu Dhabi, UAE
100044324@ku.ac.ae

Ashfaq Ahmed
C2PS center,
Khalifa University,
Abu Dhabi, UAE
ashfaq.ahmed@ku.ac.ae

Abdulhadi Shoufan
C2PS center,
Khalifa University,
Abu Dhabi, UAE
abdulhadi.shoufan@ku.ac.ae

Abstract—Outsourcing chip production is common among semiconductor vendors to cope with the increasing demand for integrated circuits. This has resulted in several security issues in the chip supply chain, including hardware trojans, intellectual property theft, and overproduction. Zero-trust presents a promising solution for ensuring the authenticity of Integrated Circuits (ICs), particularly in critical systems where adversary attacks can cause significant losses or damage. The Security Protocol and Data Model (SPDM) is a reliable protocol that uses certificates to ensure the authenticity of ICs. Based on this protocol, the presented paper proposes a chip-to-chip zero-trust security architecture that aims to verify the authenticity of any connected peripheral before its use. The contributions include an overview of the proposed architecture, implementation and formal verification of the SPDM protocol, and analysis of the challenges encountered during the implementation and execution.

Index Terms—Zero-trust, SPDM, Formal verification, SSL/TLS, Chip-to-chip communication.

I. INTRODUCTION

The world is becoming increasingly more connected at a staggering rate with the advancement of embedded systems, which are the main corps of the Internet of Things (IoT), connected cars, drones, smart homes, and industrial control systems. Embedded systems are usually based on Integrated Circuits (ICs) produced by third parties in regions with low production costs. In practice, the state-of-the-art foundries of today are considered untrusted entities in the IC supply chain, which raises some serious concerns about the trustworthiness of the fabricated ICs or devices [1]. Relying on untrusted fabrication requires a careful assessment of associated threats. For instance, any IP part of the system is inevitably shared with the untrusted foundry. Besides the obvious risks of IP theft, reverse engineering, and overproduction, the same IP can be modified by inserting a backdoor or a hardware Trojan [2]. At the same time, malicious actors are discovering more innovative ways to penetrate embedded systems through the software supply chains that build them.

New security architectures are just starting to catch up to these new threats. In the last couple of years, several new IoT-specific standards have emerged for protecting individual

IoT devices, such as the Platform Security Architecture (PSA) [3] and the Security Evaluation Standard for IoT Platforms [4]. PSA, developed by Arm, aims to provide a hardware-based isolation execution environment. It offers threat models, security analysis, and hardware/firmware specifications to create a secure foundation for IoT systems. On the other hand, the Security Evaluation Standard for IoT Platforms sets forth guidelines for evaluating the security of IoT platforms, ensuring they meet certain criteria. This standard helps manufacturers and developers assess the security features and robustness of their products, fostering a higher level of trust among users and stakeholders.

Different alternatives to combat the threats of untrusted fabrication are proposed, such as Trojan detection [5], obfuscation [6], and logic locking [7]. These last circuit-level strategies require modifications to the circuit that would render an adversary less capable of making sense of the IP. Some authors propose the reliance on trusted fabrication using split-chip solutions followed by chip-to-chip authentication [8]. These countermeasures fail at some level to provide entire secure systems. At the same time, the cybersecurity and silicon industries have recently proposed zero-trust architectures for protecting distributed infrastructure at a more granular level, especially with the next generation of open-source hardware that will surely expose a much larger attack surface to a more devastating physical world impact. The zero-trust model promises more protection in the chip supply chains [9]. The silicon industry aims to use this model to prevent the system hardware from communicating with any device that does not authenticate itself. This means that manipulations in the foundry or through the supply chain should be excluded. Furthermore, combining zero-trust security principles with existing embedded systems security approaches could help set the stage for this more robust approach to end-to-end security for the next wave of embedded systems [10]. Intel supports this model by sharing its vision and core principles towards “A Zero-Trust Approach to Architecting Silicon” [11].

In [12], a framework known as DRLGENCEART has been introduced, showcasing the utilization of deep reinforcement learning for automating the testing of certificate verification. DRLGENCEART functions by utilizing conventional certificates as input and producing novel certificates capable of efficiently highlighting discrepancies. This approach leverages deep reinforcement learning to make insightful choices during

This work was supported by the Khalifa University of Science and Technology from the External Fund under Grant 8434000388

certificate generation, informed by prior modifications, thereby enhancing the overall process.

In [13], the communication between an IoT ESP32 embedded system and an IoT Cloud is secured using MQTTS protocol with SSL/TLS certificates. The study focuses on the correctness of the encryption approach without reporting any details about the used algorithms. An architecture is proposed in [14], enabling IoT devices to authenticate data and notarize within the Ethereum blockchain. The work builds upon this concept by devising a robust hardware-software platform that empowers lightweight devices, such as IoT sensors, to orchestrate this process. Within this architecture, these devices hold a confidential key with their corresponding public address. As transactions are generated, they are seamlessly signed and dispatched to the blockchain network.

This study introduces a Chip-to-Chip Zero-Trust Architecture (C2CZTA) that is designed to ensure secure communication between two chips and includes a mechanism for verifying the authenticity of these peripherals. The foundation of C2CZTA relies on a zero-trust processor that executes the Security Protocol and Data Model (SPDM) protocol, incorporating multiple cryptographic engines for enhanced security. The primary contributions of this work encompass the compilation, optimization, and testing of the SPDM protocol. These contributions include an architectural overview, the implementation and rigorous formal verification of the SPDM protocol, and an exhaustive analysis of the implementation and execution challenges. Furthermore, the study encompasses the practical implementation of SPDM I2C communication, leveraging Raspberry Pi devices as the experimental platform.

The rest of the paper is organized as follows: Section 2 describes the proposed methods, including a general overview and the embedded implementation of the architecture. Section 3 presents and discusses the obtained experimental results. Finally, Section 4 concludes the paper and suggests future research directions.

II. METHODS

A. Architecture Overview

In this study, we propose a C2CZTA to secure communication between two chips, a requester and a responder. It should be noted that the responder represents an external peripheral, which can be an active (integrated processor) or a passive device. The responder must be authenticated to communicate with the requester; otherwise, no data exchange occurs.

Figure 1 illustrates a general overview of the proposed C2CZTA. The architecture is composed of four blocks:

- 1) Zero-Trust Processing (ZTP): This functionality is responsible for carrying out the zero-trust process and serves as the interface between the requester and the responder. Different protocols were investigated for this purpose including the PCIe IDE [15] and the SPDM [16]. The SPDM was chosen for this study because it supports non-PCIe interconnects.
- 2) Zero-Trust Management (ZTM): It determines whether the ZTP is permitted to communicate with the connected

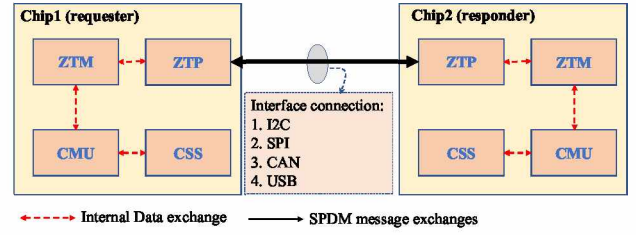


Fig. 1. Functional partitioning of zero-trust operation.

peripheral. This unit utilizes event-based scheduling. During the initialization phase and before initiating data exchange, the responder sends its certificate to the requester's ZTM to verify its validity. For instance, zero-trust activities must be logged and communicated to report a verification failure or success.

- 3) Certificate and Secret Storage (CSS): It enables saving certificates and private keys, which should be stored efficiently and securely in an on-chip flash memory.
- 4) Certificate Management Unit (CMU): It enables certificate management, such as revocation, updates, and notifications about certificate changes to other chips.

The C2CZTA can communicate with external peripherals using various interfacing protocols such as the I2C, SPI, USB, and CAN. The scope of this paper is to implement the selected authentication protocol as well as to manage the newly plugged devices. We only focus on implementing the ZTP block and CMU mechanism.

B. Zero-Trust Processing Mechanism

1) *Description of the SPDM protocol:* The SPDM protocol defines messaging formats, data objects, and sequences for secure communication between devices across various transport and physical media [16]. The SPDM includes cryptographic engines for hashing, digital signature, and verification. Furthermore, post-quantum cryptographic algorithms have recently been integrated into the SPDM protocol to make it quantum-resistant [17].

2) *Formal verification of the SPDM protocol:* The validation of a security protocol is critical before implementation. Ideally, a security protocol should only be integrated into hardware systems if it passes all formal verification tests. There are a variety of existing tools and methods for the formal verification of security protocols. In this work we use Automatic Verification of Internet Security Protocols and Applications (AVISPA) to verify the SPDM protocol for the proposed C2CZTA [18]. AVISPA is a formal verification tool with a push-button interface widely used by the research community. It consists of various backends; however, we employ an On-the-Fly Model Checker (OFMC) for verification. As described previously, the protocol includes two chips, which are also termed agents. Therefore, the tests are conducted for multiple sessions, i.e., when both chips are authorized and when one is not, with the unauthorized chip imitating an intruder. The tool supports a Dolev-Yao intruder, in which

the intruder has complete control of the network and can intercept any communication but cannot decrypt the data unless it possesses cryptographic keys. Figure 2 depicts the overall structure of the implemented formal verification. Three sessions are created within the environment, and the secrecy and authentication goals for various messages and keys are set. Furthermore, the intruder's knowledge is also defined by the environment. An illegal chip behaves as an intruder in our model. The test results demonstrate that SPDm is a secure protocol that can be embedded in hardware. For a more in-depth understanding of the formal verification process applied to the SPDm, refer to [19].

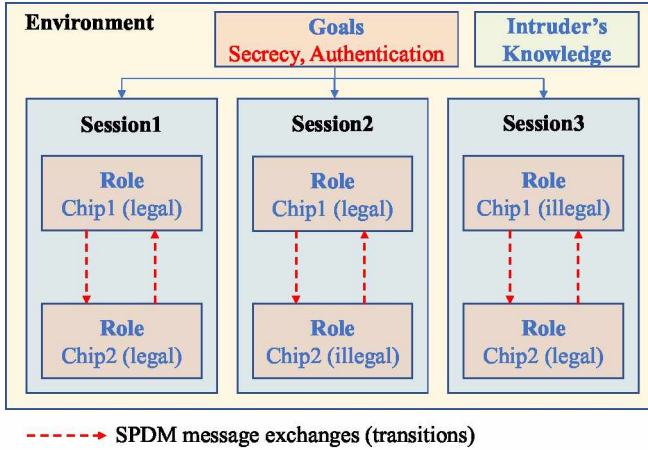


Fig. 2. An overview of implemented formal verification using AVISPA

3) *Embedded implementation of SPDm protocol*: As an illustration, the proposed architecture will be integrated into a drone to allow its flight controller to communicate only with a trusted GPS through the I2C interface. GPS modules are getting higher relevance for drone operation due to their role in remote identification [20], [21]. Figure 3 presents an overview of the prototyping environment as implemented. It consists of two Raspberry PI boards connected through the I2C bus; one configured as a requester (master) and the other as a responder (slave). The SPDm code is available on GitHub as an open-source library [22]. The current version of the library does not support any hardware interface. It only allows for the interconnection of the requester and responder via an internal socket on the same platform. The scripts are written to route communication through the I2C interface rather than the internal socket. The developed code was integrated into the SPDm library and made available to the community.

C. Certificate Management Unit Mechanism

Fig. 4 presents the different scenarios of the ZTM to deal with the plugged peripherals upon request from the ZTP. The ZTM mechanism, during the SPDm protocol's initialization phase, validates the certificates of the plugged peripherals. If a peripheral's certificate is verified, ZTM permits the ZTP to establish a communication session. This ensures that even genuine peripherals lacking proper certificates are banned from

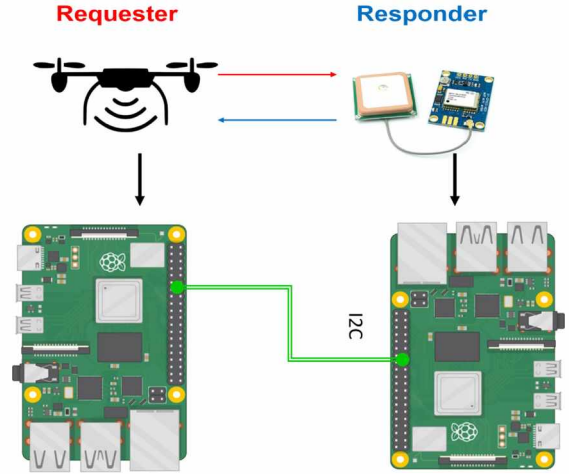


Fig. 3. Prototyping environment.

communication with the requester. To address this issue, a certification mechanism is proposed within the CMU unit to authenticate unidentified peripherals by the application administrator. The ZTM initiates a certification request to the CMU via a local agent (Domain Validation Certificate: DVC) or through a third party for approval (Extended Validation Certificate: EVC). The plugged non-certified peripherals should contain some information, such as the peripheral's manufacturing information and the Unified Identifier (UID). This information is transmitted over the Internet to the EVC, which generates and sends back corresponding certificates. Local certification can be carried out by the system administrator using tools like OpenSSL [23] or EmbedTLS [24] libraries. These libraries enable the generation of the key parameters, the creation, revocation, and update of certificates, the calculation of message digests, and the signing and verifying operations.

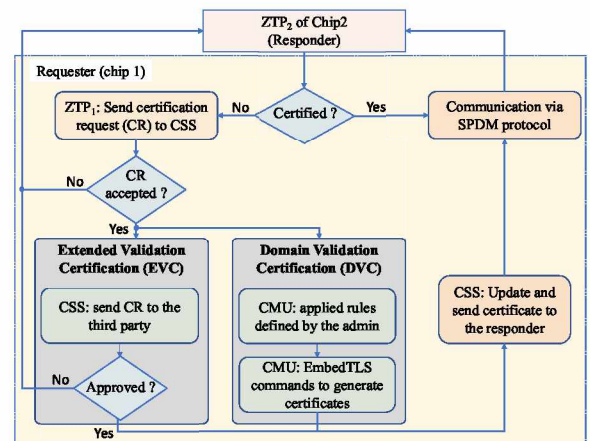


Fig. 4. Scenarios to certify genuine peripherals.

III. RESULTS AND DISCUSSION

A. Formal Verification of SPDm Protocol

Several attack scenarios such as replay attacks were tested to demonstrate the protocol's security and ability to ensure zero-trust communication between two chips. AVISPA's results indicate that the SPDm protocol is resistant to these attacks and allows for secure communication and authentication.

B. Performance Evaluation

The implementation of SPDm on Raspberry PI using I2C communication has encountered several issues, beginning with the lack of support for Raspberry PI to function as a slave. The Raspberry PI Broadcom chip BCM2711 only supports I2C master communication, creating a problem when using the Raspberry PI as a slave. One library available that can operate the Raspberry PI as a slave is the Pigiop library [25]. This library employs a FIFO buffer filled by the slave and emptied by the master. The issue is that the master does not receive the data correctly after multiple read requests; clearing the FIFO buffer before appending the data will ensure that the master receives the data in the correct order.

Table I showcases the implementation latency during the different stages of the SPDm protocol, with varying buffer sizes. The run time of the authentication procedure is around 30 seconds. The authentication procedure includes sharing the certificates, getting the user measurements, establishing a shared key, and initiating secure communications. The results show that the latency time greatly increases with an increase in the buffer size, as due to the hardware bug, for the master device to read one byte of data correctly, it needs to manually clear the buffer by sending read requests times the size of the buffer used.

TABLE I
LATENCY (IN SECOND) OF THE DIFFERENT BLOCKS OF THE SPDm PROTOCOL.

Stage	Buffer Size						
	8	16	32	64	128	256	512
Initial Communication	2.13	2.15	2.19	2.28	2.48	2.84	3.48
Get_Digest	0.73	0.76	0.81	0.93	1.18	1.67	2.47
Get_Certificate	3.49	4.06	5.08	7.48	12.05	21.89	40.79
Challenge	10.39	10.48	10.62	10.97	11.62	13.07	15.23
Key_Exchange	11.84	11.96	12.18	12.66	13.59	15.53	19.09
Finish	0.70	0.71	0.71	0.72	0.74	0.77	0.85
Total	29.28	30.12	31.59	35.04	41.66	55.77	81.91

Table II summarizes the throughput results for different sizes of the buffer. It shows that an increased buffer size significantly decreases the read and write throughput. The decrease in throughput when the buffer size increase is also due to the hardware bug, which was the same issue that impacted the latency analysis of the Implementation.

Unfortunately, the Pigiop library does not contain a function to clear the FIFO buffer. The only way to do this is to perform 512 read accesses since the default size of the buffer is 512 bytes. Therefore, whenever a master needs to read a single byte, it must send 512 read requests to clear the buffer. The solution to this delay is to reduce the buffer size to 8 bytes or less so that instead of reading 512 bytes of data each

TABLE II
THROUGHPUT (BYTE/SEC).

Operation	Buffer Size						
	8	16	32	64	128	256	512
Write	617	553	471	341	223	132	74
Read	1670	1272	908	524	289	153	80

time to receive 1 byte of data correctly, the master will only need to read 8 bytes first before receiving the correct data. This resolves the issue of clearing the buffer but imposes the restriction that the master can only read 1 byte at a time, significantly reducing the communication speed. Moreover, the data should always be appended to the FIFO buffer before the master initiates the read request, introducing another delay.

Concerning the certification process of the non-certified and genuine peripherals, it is important to be sure about the manufacturer and UID information. The system administrator must consider peripheral UID cloning as one of the major impediments. By authorizing the connection of non-certified peripherals, the system administrator assumes responsibility.

C. Benchmarking the Complete Framework

Table III compares the proposed system with other embedded authentication protocol implementations. It presents various evaluation criteria, including the security domain, focus, methodology, protocol/model, and evaluation platform.

The proposed work focuses on securing the chip supply chain at the hardware layer, addressing issues such as hardware trojans, intellectual property theft, and overproduction. It introduces C2CZTA that verifies the authenticity of connected peripherals, mitigating risks associated with compromised or counterfeit chips. The work utilizes the SPDm protocol for authentication, specifically designed for the hardware layer's security requirements. The Raspberry Pi evaluation platform based on ARM architecture demonstrates the practicality of implementing C2CZTA in real-world scenarios using widely available embedded systems. In contrast, the works presented in [12]–[14] concentrate on certificate verification, generation, or data certification on the blockchain.

The C2CZTA offers notable security enhancements for chip communication. However, integrating C2CZTA into existing systems or devices might be challenging due to compatibility issues with legacy hardware or software. Furthermore, latency in authentication and increased resource consumption might affect system performance. These limitations can be managed through careful design and optimization.

IV. CONCLUSION

In this study, we proposed a C2CZTA to secure the communications between chips. The proposed architecture incorporates the SPDm protocol, which enables data communications with confidentiality and integrity protection. Using AVISPA tools, SPDm is subjected to formal verification to determine the protocol's correctness and resilience to various attack scenarios. As a case study, the protocol is implemented on two Raspberry PI to demonstrate that the two platforms can

TABLE III
COMPARISON OF THE PROPOSED SYSTEM WITH OTHER EMBEDDED AUTHENTICATION PROTOCOLS IMPLEMENTATION.

Criteria	The proposed work	[12]	[13]	[14]
Security Domain	Hardware layer	Transport layer	Transport layer	Application, transport, and network layers
Focus	Chip supply chain	Certificate verification in SSL/TLS	Certificate generation in SSL/TLS	Data certification and notarization on blockchain
Methodology	Chip-to-chip zero-trust architecture	Deep reinforcement learning	Automated certificate generation	Ethereum blockchain
Protocol/Model	SPDM	SSL/TLS	SSL/TLS	data certification on blockchain
Evaluation platform	Raspberry PI based on ARM	Laptop based on Intel	ESP32	ARM Cortex-M4

communicate once the certificates are validated. A certification mechanism is established to generate new credentials for new genuine and non-certified peripherals.

Our future work will focus on developing an embedded system-on-chip (SoC) based on a RISC-V processor that integrates the various blocks of the proposed architecture. Also, light-weight accelerators for traditional and post-quantum cryptographic algorithms will be considered [26] to speed up the authentication and secure communication between chips for the zero-trust era.

REFERENCES

- [1] E. Prouff and P. Schaumont, Eds., *Cryptographic Hardware and Embedded Systems – CHES 2012*. Springer Berlin Heidelberg, 2012. [Online]. Available: <https://doi.org/10.1007/978-3-642-33027-8>
- [2] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.
- [3] J. Jung, J. Cho, and B. Lee, "A secure platform for IoT devices based on ARM platform security architecture," in *International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 2020, pp. 1–4.
- [4] "IEEE standard for wireless diabetes device security assurance evaluation: Connected electronic product security evaluation programs," *IEEE Std 2621.1-2022*, pp. 1–22, 2022.
- [5] A. Jain, Z. Zhou, and U. Guin, "Survey of recent developments for hardware trojan detection," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5.
- [6] N. Rangarajan, S. Patnaik, J. Knechtel, R. Karri, O. Sinanoglu, and S. Rakheja, "Opening the doors to dynamic camouflaging: Harnessing the power of polymorphic devices," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 1, pp. 137–156, 2022.
- [7] Y. Zhong and U. Guin, "Complexity analysis of the SAT attack on logic locking," 2022. [Online]. Available: <https://arxiv.org/abs/2207.01808>
- [8] I. Karageorgos, M. M. Isgenc, S. Pagliarini, and L. Pileggi, "Chip-to-chip authentication method based on SRAM PUF and public key cryptography," *Journal of Hardware and Systems Security*, vol. 3, no. 4, pp. 382–396, Nov. 2019. [Online]. Available: <https://doi.org/10.1007/s41635-019-00080-y>
- [9] A. Varas, R. Varadarajan, J. Goodrich, and F. Yinug, "Strengthening the global semiconductor value chain," Semiconductor Industry Association (SIA)/Boston Consulting Group (BCG), Tech. Rep., April 2021. [Online]. Available: https://www.semiconductors.org/wp-content/uploads/2021/05/BCG-x-SIA-Strengthening-the-Global-Semiconductor-Value-Chain-April-2021_1.pdf
- [10] C. Conlon and C. Garlati, "A new zero-trust model for securing embedded systems," in *Embedded World Conference 2019 Proceedings*, March 2019. [Online]. Available: <https://bringyourownit.com/2019/03/03/a-zero-trust-model-for-securing-embedded-systems/>
- [11] M. G. Dixon, "A zero trust approach to architecting silicon," Intel, Tech. Rep., accessed: 2022-10-24. [Online]. Available: <https://www.intel.com/content/www/us/en/newsroom/opinion/zero-trust-approach-architecting-silicon.html#gs.48k7bp>
- [12] C. Chen, W. Diao, Y. Zeng, S. Guo, and C. Hu, "Drlgencert: Deep learning-based automated testing of certificate verification in SSL/TLS implementations," in *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2018, pp. 48–58.
- [13] N. Nikolov and O. Nakov, "Research of secure communication of Esp32 IoT embedded system to .NET core cloud structure using MQTTS SSL/TLS," in *IEEE XXVIII International Scientific Conference Electronics (ET)*, 2019, pp. 1–4.
- [14] G. Rafaiani, P. Santini, M. Baldi, and F. Chiaraluce, "Implementation of ethereum accounts and transactions on embedded IoT devices," in *IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, 2022, pp. 1–6.
- [15] D. Das Sharma, "Keynote 1: Compute express link (CXL) changing the game for cloud computing," in *IEEE Symposium on High-Performance Interconnects (HOTI)*, 2021, pp. xii–xii.
- [16] "DSP0277 : Secured messages using SPDM specification, version: 1.1.0 specification," DMTF, Tech. Rep. 1.1.0, 2022, accessed: 2022-10-24. [Online]. Available: https://www.dmtf.org/sites/default/files/standards/documents/DSP0277_1.1.0.pdf
- [17] J. Yao, K. Matusiewicz, and V. Zimmer, "Post quantum design in SPDM for device authentication and key establishment," *Cryptography*, vol. 6, no. 4, p. 48, sep 2022. [Online]. Available: <https://doi.org/10.3390/cryptography6040048>
- [18] L. Viganò, "Automated security protocol analysis with the AVISPA tool," *Electronic Notes in Theoretical Computer Science*, vol. 155, pp. 61–86, 2006.
- [19] A. Ahmed, A. Shoufan, and K. Belwafi, "Light-weight security protocol and data model for chip-to-chip zero-trust," *IEEE Access*, vol. 11, pp. 60 335–60 348, 2023. [Online]. Available: <https://doi.org/10.1109/access.2023.3285630>
- [20] K. Belwafi, R. Alkadi, S. A. Alameri, H. Al Hamadi, and A. Shoufan, "Unmanned aerial vehicles' remote identification: A tutorial and survey," *IEEE Access*, vol. 10, pp. 87 577–87 601, 2022.
- [21] S. Alshamsi, M. Y. Alhashmi, K. Belwafi, and A. Shoufan, "A low-power remote identification module for drones," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2023, pp. 1–5.
- [22] DMTF, "libspdm," <https://github.com/DMTF/libspdm>, accessed: 2022-10-24.
- [23] J. Walden, "OpenSSL 3.0.0: An exploratory case study," in *IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*, 2022, pp. 735–737.
- [24] M. Lipp, A. Kogler, D. Oswald, M. Schwarz, C. Easdon, C. Canella, and D. Gruss, "PLATYPUS: Software-based power side-channel attacks on x86," in *IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 355–371.
- [25] M. Nasir, "How to install pigpio on raspberry pi." [Online]. Available: <https://linuxhint.com/install-pigpio-raspberry-pi/>
- [26] R. Laue, O. Kelm, S. Schipp, A. Shoufan, and S. A. Huss, "Compact aes-based architecture for symmetric encryption, hash function, and random number generation," in *2007 International Conference on Field Programmable Logic and Applications*. IEEE, 2007, pp. 480–484.