POLITECNICO DI TORINO
Repository ISTITUZIONALE

Securing IoT Devices: an Overview

Original
Securing IoT Devices: an Overview / Bravi, E., Lioy, A.. - ELETTRONICO. - (2025). (2025 IEEE Symposium on Computers and Communications (ISCC) Bologna (ITA) 2-5 July 2025) [10.1109/ISCC65549.2025.11326324].

Availability:
This version is available at: 11583/3000310 since: 2026-01-15T15:11:27Z

Publisher:
IEEE

Published
DOI:10.1109/ISCC65549.2025.11326324

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

18 November 2026

Securing IoT Devices: An Overview

Enrico Bravi
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy
enrico.bravi@polito.it

Antonio Lioy
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy
antonio.lioy@polito.it

***Abstract*—IoT devices are becoming increasingly popular. However, they are vulnerable to several security attacks because of their resource-constrained nature, making it challenging to protect them with traditional security countermeasures. To cope with the resource limitations of these devices, researchers have proposed ad-hoc versions of classical security controls, such as cryptography and hardware root-of-trust. Lightweight cryptography focuses on developing efficient cryptographic algorithms regarding required memory and processing power. CBOR X.509 certificates are a lightweight and secure way to represent X.509 certificates. They are significantly smaller than traditional DER-encoded certificates and can be encoded and decoded more efficiently. This makes them well-suited for use in IoT devices, where resources are often limited. Remote Attestation (RA) is a security mechanism that permits a trusted party to verify that a platform behaves as expected. RA techniques are generally not suitable for constrained devices, as they require additional hardware components or extensions. Recently, several proposals have been proposed to provide similar security capabilities to devices with very low computational resources. This can be used to detect and prevent malicious devices from accessing IoT networks. This paper analyses some of these new proposals, technologies, and possible integrations to create secure and efficient IoT systems.**

***Index Terms*—ASCON, C509, CBOR, Cybersecurity, DICE, Internet of Things, MARS, Remote Attestation**

I. Introduction

The Internet of Things (IoT) [1] is a network of interconnected devices, from simple sensors to complex systems. IoT devices are used in many applications [2], including smart homes, healthcare, and industrial automation. However, the increasing prevalence of IoT devices also introduces new security challenges [3]. One of these challenges for IoT devices is their resource-constrained nature, which makes them difficult to protect with traditional security measures. Due to the insecure nature of these devices, several attacks can be performed to steal sensitive information, such as encryption keys and passwords, or to tamper with a device. Researchers are developing new security mechanisms and technologies for IoT devices to address these security challenges.

Cryptography is an important security control, but it is often a critical task on IoT devices regarding execution time and resource consumption. Lightweight cryptography focuses on developing efficient cryptographic algorithms in terms of memory usage and processing power. This makes it well-suited for use in resource-constrained IoT devices. Some competitions have been proposed to select algorithms for constrained devices. The competition launched by the National Institute of Standards and Technology (NIST) selected [4] the ASCON [5] family of algorithms, which provide solutions for Authenticated Encryption with Associated Data (AEAD) and hash calculation.

Researchers and standardisation organisations are working to enable, on IoT devices, protocols and procedures that can be considered a standard *de facto* in more powerful contexts.

The C509 [6] is an IETF draft for a lightweight representation of X.509 certificates. C509 certificates are significantly smaller than traditional DER-encoded X.509 certificates and can be encoded and decoded more efficiently. This makes them well-suited for use in IoT, where bandwidth and memory capacity are often limited.

Remote Attestation (RA) is a procedure to remotely verify a device's software and hardware integrity. In the case of IoT, this becomes challenging because not all devices satisfy the resource requirements to perform RA. Proposals have been made, but some of them are software-based solutions [7]–[9] that offer fewer guarantees than hardware-based solutions. For this reason, RA is a process mostly used in cloud computing, or more generally, where the involved devices have the necessary resources, like external secure elements or hardware extensions [10]. More recently, some works have been published that move to a hardware-based approach [11]–[14]. Along this line, new standard proposals are emerging, such as the Device Identifier Composition Engine (DICE) [15] and Measurement and Attestation RootS (MARS) [16].

In this paper, we analyse some of these new technologies, explain their details, and propose some integration to increase security for IoT.

II. Lightweight Cryptography

In the ever-evolving landscape of cryptography, *Lightweight Cryptography* [17] is designed to address the specific security needs of IoT devices, which are often resource-constrained. These devices typically have limited computational power, memory, and energy resources. Traditional cryptographic algorithms can be overly onerous for such devices, making them unsuitable for IoT applications. The primary goal of

This work was partially supported by the project SERICS (PE00000014) under the NRRP MUR program, funded by the European Union - NextGenerationEU. This publication is also part of the project PNRR-NGEU which has received funding from the MUR - DM 352/2022.

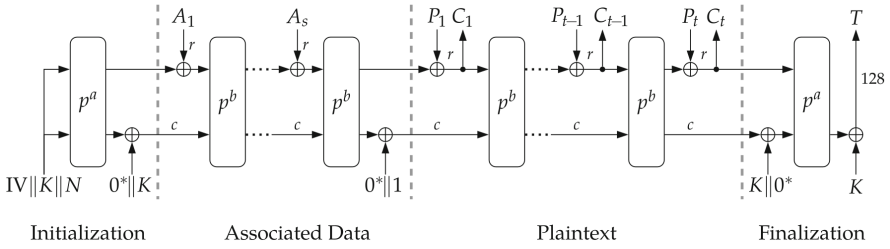


Fig. 1. ASCON encryption schema (source: [5]).

Lightweight Cryptography is to provide cryptographic algorithms that strike a balance between security and efficiency.

In this context, in 2014, a CESAR [18] competition for "Lightweight applications (resource-constrained environments)" was started. The selected algorithms for this use case were, as the first choice, ASCON [5] and ACRON [19] as a second choice. The ASCON family was later chosen for standardisation by the NIST for authenticated encryption and hashing algorithms.

A. ASCON

ASCON is a lightweight cryptographic family of algorithms designed for efficiency in constrained environments, such as embedded systems. It provides encryption and authentication, making it an AEAD [20] cipher, and the possibility to compute hashes based on a sponge operation [21], similar to SHA-3 [22]. An AEAD cipher permits the obtaining of confidentiality and data authentication at the same time. In addition, it permits covering the integrity of property and data that does not need to be confidential (associated data). This is very useful, for example, to protect a network packet, where the payload can be encrypted, but the header cannot because it contains information for the correct packet routing.

The ASCON crucial function is the *permutation* function. It takes a 320-bit state as input and processes it through two fixed numbers of rounds p^a (typically 12) and p^b (typically 6). Each round comprises three distinct steps:

1) *Addition of Constants*. Unique values specific to each round are added to the state to introduce diffusion and confusion. This is the initial step in breaking down statistical patterns and enhancing security.
2) *Substitution Layer*. A non-linear S-box [5] operation is applied to each bit in the state. This crucial step introduces non-linearity, making it significantly harder for an attacker to predict the output from the input, thereby bolstering resistance to differential attacks.
3) *Linear Diffusion Layer*. A bit mixing operation, known as the Linear Layer, diffuses the information further across the state. This step ensures that every bit in the output depends on multiple bits in the input, strengthening the overall security against various cryptanalytic techniques.

The combined effect of these operations within each round transforms the state, ultimately producing the final output after completing all rounds.

The encryption (Fig. 1) and decryption procedures are performed in four steps:

1) *Initialisation*: The initial state is derived from the key, nonce, and a constant. This state is then permuted using the Ascon permutation.
2) *Associated Data Processing*: Any associated data (optional) is absorbed into the state in blocks, with the state being permuted after each block.
3) *Plaintext Processing*: The plaintext is encrypted by XORing it with a part of the state, producing the ciphertext. The state is updated and permuted after each block.
4) *Finalisation*: After all plaintext has been processed, the remaining state is permuted one last time and then truncated to produce the authentication tag.

For AEAD, the block size is $64\,\mathrm{bit}$ long but can also be $128\,\mathrm{bit}$, increasing the number of rounds p^b to 8. The key and the nonce are $128\,\mathrm{bit}$ bits long, and the generated tag.

III. CBOR Encoded X.509 Certificates (C509 certificates)

Concise Binary Object Representation (CBOR) [23] is a binary data format designed to be efficient and extensible, similar to other data representations such as JSON and XML. The main goal of CBOR is to have a very small representation size and extensibility without the need for version negotiation.

Here are listed the fields used by the CBOR encoding:

- *Data Types*: CBOR has several data types, including unsigned integers, negative integers, byte strings, text strings, arrays, maps (key-value pairs), tags, simple values (like true, false, null), and floating point numbers. This concept is mapped into prefixes placed before the actual value.
- *Prefixes*: Each data item starts with a byte that includes both the major type (the high-order 3 bits) and additional information (the low-order 5 bits). The major type indicates the data item type (unsigned integer, string, array, etc.). The additional information can either provide the actual value for small enough integers or the length of the data item.
- *Lengths and Values*: For larger data items, the length or value is encoded in the bytes following the initial byte (prefix). The number of bytes used depends on the value of the additional information in the initial byte.

- *Arrays and Maps*: Arrays and maps start with an initial byte indicating the type and the number of items, followed by the encoding of each item.
- *Tags*: Tags are a way to add optional metadata to a data item. A tag is a data item itself, followed by the data item it applies to.
- *Indefinite Lengths*: CBOR supports encoding data items of unknown length, such as a byte string streaming over a network. These data items are marked with a special "indefinite length" marker (`0x1F`, 31 in decimal representation, put in the lower 5 bit of the prefix) and end with a "break" marker (`0xFF` corresponding to 7 represented in the higher 3 bit and 31 in the lower 5 bit).

Here's a simple example of CBOR encoding. The map to encode is the following:

`{"A": 1, "B": 2}` (13 bytes)

The initial byte is `0xA2` where `0xA` is the major type for a map, and `2` is the number of items. The first key is the string "A". It's encoded as `0x61 0x41` where `0x61` is the initial byte, with `0x6` indicating a string and `1` the length. `0x41` is the ASCII value for "A". The first value is the integer `1`. It's encoded as `0x01`, with `0x0` indicating an unsigned integer and `1` the value. The second key-value pair is encoded similarly, resulting in `0x61 0x42 0x02`. Putting it all together, the entire CBOR encoding is:

`0xA2 0x61 0x41 0x01 0x61 0x42 0x02`

(7 bytes)

This representation permits encoding a 13 byte JSON map to a 7 byte bitstream, noticeably reducing the dimension of representation.

C509 certificates [6] are encoded using the CBOR data format. C509 certificates are significantly smaller than traditional DER-encoded X.509 certificates and can be encoded and decoded more efficiently. An example of CBOR compression is reported in Fig. 2. In this example, the Issuer field is reported in the standard ASN.1 [24] encoding, and then it is compared with the CBOR compression obtained.

This makes C509 certificates suitable for use in applications where bandwidth is limited, such as mobile devices and IoT devices. C509 certificates can be used in the same way as traditional DER-encoded X.509 certificates. For example, they can authenticate servers and clients in TLS connections, devices in IoT networks, and sign codes and documents.

To be compatible with existing processes, for example, used by Certificate Authorities (CAs) for issuing certificates, the possibility to manage C509 certificates can be integrated without managing the C509 format natively. In this case, it is sufficient that the CA implements an upper logical layer, where an incoming C509 certificate of Certificate Signing Request (CSR) is converted into DER format. In this way, changing the already implemented procedure would be unnecessary because all operations would be performed in the DER format.

```
-- Uncompressed ASN.1 --
0x30    // Sequence
0x12    // Size 18
0x31    // Set
0x10    // Size 16
0x30    // Sequence
0x0E    // Size 14
0x06    // OID
0x03    // Size 3
0x55 0x04 0x03
        // 2.5.4.3
            (commonName)
0x0C    // UTF8 string
0x07    // Size 7
0x52 0x6F 0x6F 0x74
    0x20 0x43 0x41
        // Value "Root CA"

    (20 bytes)
```

```
-- Compressed CBOR --
0x67    // Text string of size 7
0x52 0x6F 0x6F 0x74
    0x20 0x43 0x41
        // Value "Root CA"

    (8 bytes)
```

Fig. 2. Example CBOR Encoding (Issuer).

The adoption of this format for IoT devices would permit to provide them the security properties of X.509 certificates, but taking into consideration the resource constraints that affect this family of devices. Of course, it remains the fact that asymmetric encryption is still very heavy in terms of computational capabilities. Still, this introduction would be a crucial improvement, considering, for example, the new protocols designed, such as the Constrained Application Protocol (CoAP) [25], which introduces the possibility of implementing an HTTP-compatible protocol in a lightweight environment.

IV. Remote Attestation in IoT Scenario

The possibility of verifying the trustworthiness of platforms is essential for several critical scenarios, where knowing that the hardware and software components are not tampered with is crucial. This can be achieved with Remote Attestation (RA) [26], [27]. RA is a security procedure that allows a trusted platform, known as the *Verifier*, to confirm the integrity of another platform, referred to as the *Attester*, by utilising predefined trusted values stored in the Verifier, which correspond to the expected state of the Attester. The RA process is generally defined as a challenge-response protocol (Fig. 3), where the Verifier sends a challenge to an Attester to obtain its current state. The Attester computes proof of its internal state and sends it back to the Verifier. In this way, the Verifier can determine the trustworthiness of the Attester, verifying that the proof received corresponds to the expected state.

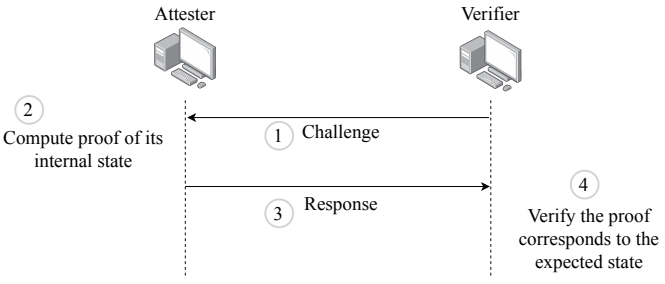


Fig. 3. Basic RA process.

Numerous RA methods have been suggested [28], [29], each relying on various hardware and software technologies to safeguard sensitive data, like cryptographic keys, employed in the procedure.

a) *Hardware-Based Attestation*

These techniques centre around specialised hardware components, typically a dedicated cryptographic chip that must be present on the Prover. The Trusted Platform Module (TPM) is one of the most widely used hardware devices for conducting RA. The Trusted Computing Group (TCG) initially introduced the TPM specification, with the initial version being 1.2 [30]. Subsequently, an improved version, TPM 2.0 [31], became the prevailing standard. Other RA approaches leverage a Trusted Execution Environment (TEE) [32], such as Intel SGX [33] or ARM TrustZone [34]. These methods often rely on specific hardware extensions.

b) *Software-Based Attestation*

Although hardware-based attestation is a highly effective RA solution, it may not always be feasible due to hardware and software limitations, especially in embedded devices. To address this challenge, some RA approaches that are solely based on software have been developed to reduce hardware overhead. An example is Pioneer [8], which is a software-based primitive that does not depend on CPU architecture extensions or secure co-processors.

c) *Hybrid Attestation*

While software-based RA methods may not be sufficient in some network environments due to potential adversary capabilities [35], a hybrid approach has been devised to tackle this issue, combining both software and hardware elements. SMART [36] serves as an example of this hybrid approach, involving minimal hardware modifications to embedded Micro Controller Units.

An issue for the IoT scenario is the lack of a secure coprocessor that represents the Root of Trust (RoT) for performing RA. The RoT is a component that is considered trusted, so it is expected to behave correctly because some misbehaviour cannot be detected at runtime. In a Trusted Platform [38], three RoT are required by the TCG:

1) *Root of Trust for Storage* (RTS): it provides secure storage where to store measurements of the platform components;
2) *Root of Trust for Measurement* (RTM): it has the purpose of measuring all the platform components and sending these values into the RTS;
3) *Root of Trust for Reporting* (RTR): it has the purpose of securely reporting the content of RTS.

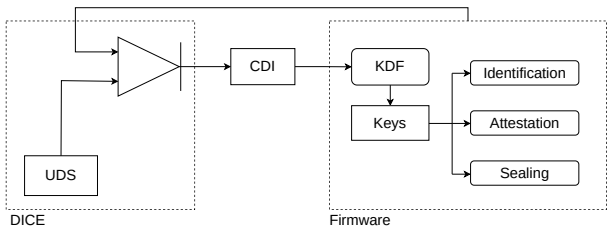


Fig. 4. DICE architecture (source [37]).

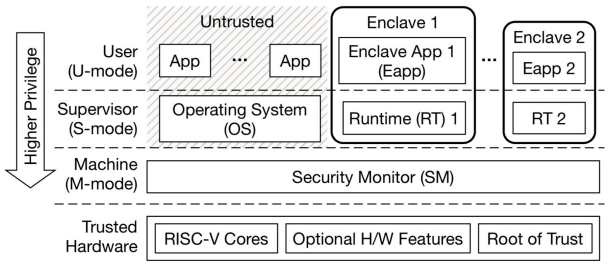


Fig. 5. Keystone architecture (source: [39]).

A. Device Identifier Composition Engine (DICE)

To minimise the hardware requirements for a RoT, the TCG proposed the Device Identifier Composition Engine (DICE) [15] specification. DICE permits the establishment of a RoT and provides device identity based on the device's integrity state. This permits the strict linking of the identity of the device to its code and configuration. The DICE, which is a shielded region (e.g. ROM), possesses a manufacturer-inserted value, the Unique Device Secret (UDS). This value is used during the device boot to derive the Compound Device Identifier (CDI) associated with the device's code. This value is derived using a one-way function (OWF), which can be used by the firmware that receives it to derive, for example, its cryptographic keys with a Key Derivation Function (KDF). The mechanism used by DICE for deriving all the cryptographic material for the device is shown in Fig. 4.

A different approach proposed to protect IoT applications is Keystone [39], which is an open-source TEE proposed by Lee *et al.*, which is based on the Physical Memory Protection (PMP) security extension for the RISC-V ISA [40]. This framework permits the creation of several protected and isolated environments (enclaves) where it is possible to run sensitive applications. The architecture of this framework is multi-layer (Fig. 5). The first layer represents the hardware, so a RISC-V chip with the PMP extension. The second layer is the Security Monitor (SM), which is the most privileged component and is in charge of managing the life cycle of the enclaves. The last layer is represented by the operating system and its applications, for the untrusted part, the unprotected portion of the system. In the last layer are also present the enclaves, that run protected by the PMP feature set by the SM during the enclave instantiation phase. An enclave comprises the enclave application, which is the application to protect, and the Runtime, which permits the application to interact with the hardware or the SM.

Analysing these two proposals, it is possible to integrate them to merge the security advantages provided by both. The DICE specification can be used to provide a RoT to Keystone, allowing the possibility of giving an enclave an identity strictly related to the hardware platform. This can be easily applied

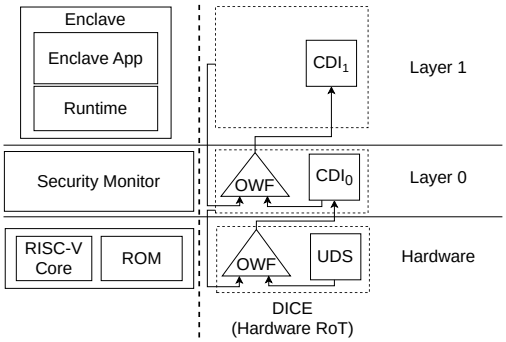


Fig. 6. Parallelism between DICE end Keystone architectures.

because DICE, as it has been defined, can be applied to multi-layer architecture and, as shown in Fig. 6, Keystone has several software layers that compose its architecture. This kind of integration would introduce several security properties, for example, it would not be possible to migrate a specific enclave application because the identity values are strictly linked to the hardware underneath. In addition, it would permit the detection of manipulation of the enclave applications because a modification would imply a change in the CDI related to it because the CDI is directly calculated from the code implementing the enclave application.

B. Measurement and Attestation RootS (MARS)

The TCG has also recently proposed a new architecture: the MARS specification [16]. This new proposal aims to provide hardware RoT specifications for IoT devices, particularly for very constrained devices. In an IoT device architecture, MARS represents a RoT, in particular, it implements the RTS and RTR. It follows the Trusted Platform definition already proposed concerning the TPM 2.0 specification [42], and it can be implemented as a discrete chip, attached to the platform and communicating through an external bus. It can be implemented as an integrated System on Chip logical block (Fig. 7) or, eventually, as a logical block that allows execution only in a privileged mode.

From a cryptographic point of view, MARS has to implement a hash function to be able to compute and store digests. This is needed because MARS, similarly to the TPM, possesses some (at least one) Platform Configuration Registers (PCRs), which can securely store the events recorded on the platform. Those PCRs are also used to report the events that happened on the platform because they possess the system's history. A PCR, as defined in the MARS specification, can be written only following the *extend* operation:

$$PCR_{new} = \textbf{Hash}_{Algorithm}(PCR_{old} \mid\mid digest)$$

where the new PCR value is calculated by concatenating the current PCR value with the digest that has to be stored and then hashing this concatenation. This procedure permits maintaining all the history of extensions and, of course, maintaining also a log to have access to all intermediate values.

The TCG explicitly mentions [16] that a TPM should be able to verify, for example, a MARS signature, for this reason, MARS should implement cryptography modules compatible with the TPM specification. It is also mentioned that MARS, for its minimalist approach, should implement a single cryptography module, but it is not mandatory, so in case of necessity, MARS can implement several cryptographic algorithms.

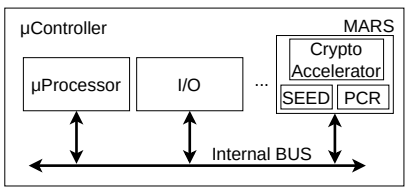


Fig. 7. Trusted Platform Architecture with MARS (source [41]).

V. Conclusion

Security is crucial in digital scenarios, from personal devices to IT infrastructures. Several solutions have been developed to protect devices, such as secure transmission protocols, cryptographic algorithms, and integrity verification procedures. IoT devices brought new challenges regarding their protection because most security solutions are unsuitable for these devices, due to their resource constraints.

Nowadays, the work in these areas is focused on developing new techniques and more efficient algorithms for constrained devices. The ASCON algorithm family was chosen by NIST for lightweight cryptography standardization. ASCON is a set of lightweight cryptographic algorithms that have been designed to be secure, efficient, and easy to implement. These algorithms are well-suited for resource-constrained devices.

Another recent proposal is the C509 certificate representation, which proposes a new and innovative way to encode X.509 certificates by using the Concise Binary Object Representation (CBOR). This offers several benefits over traditional DER-encoded X.509 certificates, including smaller size, faster encoding and decoding, and extensibility.

Also, the Trusted Computing Group is developing some proposals directly oriented to IoT devices. In this context, the Device Identifier Composition Engine (DICE) and the Measurement and Attestation RootS (MARS) specifications are the most relevant ones. These two specifications are very different from the implementation point of view. Still, they try to provide the security capabilities generally provided by a hardware Root of Trust on Iot devices.

By combining these new solutions, it would be possible to create secure and efficient IoT devices that can be used in a wide range of applications, with support for cryptographic operations, storing and manipulating X.509 certificates, and integrity verification.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey", Elsevier Computer Networks, vol. 54, June 2010, pp. 2787–2805, DOI 10.1016/j.comnet.2010.05.010

[2] A. Khanna and S. Kaur, "Internet of Things (IoT), Applications and Challenges: A Comprehensive Review", Wireless Personal Communications, vol. 114, September 2020, pp. 1687–1762, DOI 10.1007/s11277-020-07446-4

[3] J. Mohanty, S. Mishra, S. Patra, B. Pati, and C. R. Panigrahi, "IoT Security, Challenges, and Solutions: A Review", Progress in Advanced Computing and Intelligent Engineering, Singapore, November 2020, pp. 493–504, DOI 10.1007/978-981-15-6353-9_46

[4] NIST, "Lightweight Cryptography Standardization Process: NIST Selects Ascon", February 2023, https://csrc.nist.gov/news/2023/lightweight-cryptography-nist-selects-ascon

[5] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon v1.2: Lightweight Authenticated Encryption and Hashing", Journal of Cryptology, vol. 34, June 2021, DOI 10.1007/s00145-021-09398-9

[6] J. P. Mattsson, G. Selander, S. Raza, J. Höglund, and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", March 2025, https://datatracker.ietf.org/doc/draft-ietf-cose-cbor-encoded-cert/13/

[7] A. Seshadri, A. Perrig, L. Van Doom, and P. Khosla, "SWATT: SoftWare-based ATTestation for embedded devices", IEEE Symposium on Security and Privacy, Berkeley (CA, USA), May 2004, p. 272–282, DOI 10.1109/SECPRI.2004.1301329

[8] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla, "Pioneer: Verifying Code Integrity and Enforcing Untampered Code Execution on Legacy Systems", 20th ACM Symposium on Operating Systems Principles, Brighton (United Kingdom), October 2005, p. 1–16, DOI 10.1145/1095810.1095812

[9] Y.-G. Choi, J. Kang, and D. Nyang, "Proactive Code Verification Protocol in Wireless Sensor Network", Computational Science and Its Applications – ICCSA, Kuala Lumpur (Malaysia), August 2007, pp. 1085–1096, DOI 10.1007/978-3-540-74477-1_97

[10] E. Bravi, D. G. Berbecaru, and A. Lioy, "A Flexible Trust Manager for Remote Attestation in Heterogeneous Critical Infrastructures", 2023 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Naples (Italy), December 2023, pp. 91–98, DOI 10.1109/CloudCom59040.2023.00027

[11] J. Kong, F. Koushanfar, P. K. Pendyala, A.-R. Sadeghi, and C. Wachsmann, "PUFatt: Embedded Platform Attestation Based on Novel Processor-Based PUFs", 51st Annual ACM Design Automation Conference, San Francisco (CA, USA), June 2014, p. 1–6, DOI 10.1145/2593069.2593192

[12] W. Feng, Y. Qin, S. Zhao, and D. Feng, "AAoT: Lightweight attestation and authentication of low-resource things in IoT and CPS", Elsevier Computer Networks, vol. 134, December 2018, pp. 167–182, DOI 10.1016/j.comnet.2018.01.039

[13] M. N. Aman and B. Sikdar, "ATT-Auth: A Hybrid Protocol for Industrial IoT Attestation With Authentication", IEEE Internet of Things Journal, vol. 5, December 2018, pp. 5119–5131, DOI 10.1109/JIOT.2018.2866623

[14] E. Bravi, S. Sisinni, and A. Lioy, "Exploiting the DICE specification to ensure strong identity and integrity of IoT devices", 8th International Conference on Smart and Sustainable Technologies (SpliTech), Split/Bol (Croatia), June 2023, DOI 10.23919/SpliTech58164.2023.10193517

[15] TCG, "Hardware Requirements for a Device Identifier Composition Engine", March 2018, https://trustedcomputinggroup.org/wp-content/uploads/Hardware-Requirements-for-Device-Identifier-Composition-Engine-r78_For-Publication.pdf

[16] TCG, "Measurement and Attestation RootS (MARS) Library Specification", January 2023, https://trustedcomputinggroup.org/wp-content/uploads/TCG_MARS_Library_Spec_v1r14_pub.pdf

[17] S. L. William J. Buchanan and R. Asif, "Lightweight cryptography methods", Journal of Cyber Security Technology, vol. 1, March 2018, pp. 187–201, DOI 10.1080/23742917.2017.1384917

[18] CESAR, "Cryptographic competitions", https://competitions.cr.yp.to/caesar-submissions.html

[19] H. Wu, "ACORN: A Lightweight Authenticated Cipher (v3)", 2015, https://competitions.cr.yp.to/round3/acornv3.pdf

[20] P. Rogaway, "Authenticated-encryption with associated-data", 9th ACM conference on Computer and communications security (CCS '02), Washington (DC, USA), November 2002, p. 98–107, DOI 10.1145/586110.586125

[21] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, "Sponge functions", Ecrypt Hash Workshop, 2007, pp. 295–301. https://keccak.team/files/SpongeFunctions.pdf

[22] M. Dworkin, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", NIST FIPS-202, August 2015, DOI 10.6028/NIST.FIPS.202

[23] C. Bormann and P. E. Hoffman, "Concise Binary Object Representation (CBOR)", RFC-8949, December 2020, DOI 10.17487/RFC8949

[24] I. T. Union, "X.680: Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", 2008, https://www.itu.int/rec/T-REC-X.680/

[25] C. Bormann, S. Lemay, H. Tschofenig, K. Hartke, B. Silverajan, and B. Raymor, "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC-8323, February 2018, DOI 10.17487/RFC8323

[26] G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. O'Hanlon, J. Ramsdell, A. Segall, J. Sheehy, and B. Sniffen, "Principles of remote attestation", International Journal of Information Security, vol. 10, April 2011, pp. 63–81, DOI 10.1007/s10207-011-0124-7

[27] A. S. Banks, M. Kisiel, and P. Korsholm, "Remote attestation: a literature review", May 2021, DOI 10.48550/arXiv.2105.02466

[28] M. Ambrosin, M. Conti, A. Ibrahim, G. Neven, A. Sadeghi, and M. Schunter, "SANA: secure and scalable aggregate network attestation", Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, eds.), Vienna (Austria), October 24-28 2016, pp. 731–742, DOI 10.1145/2976749.2978335

[29] L. Ferro, E. Bravi, S. Sisinni, and A. Lioy, "SAFEHIVE: Secure Attestation Framework for Embedded and Heterogeneous IoT Devices in Variable Environments", 2024 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems (SaT-CPS), Porto (Portugal), 2024, p. 41–50, DOI 10.1145/3643650.3658609

[30] TCG, "Trusted Computing Group Protection Profile PC Client Specific Trusted Platform Module TPM Family 1.2", July 2008, https://www.commoncriteriaportal.org/files/ppfiles/pp0030b.pdf

[31] TCG, "Protection Profile PC Client Specific TPM Library Specification Family 2.0", September 2021, https://trustedcomputinggroup.org/wp-content/uploads/TCG_PCClient_PP_1p3_for_Library_1p59_pub_29sept2021.pdf

[32] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted Execution Environment: What It is, and What It is Not", IEEE Trustcom/BigDataSE/ISPA, Helsinki (Finland), August 2015, pp. 57–64, DOI 10.1109/Trustcom.2015.357

[33] V. Costan and S. Devadas, "Intel SGX explained", 2016, https://eprint.iacr.org/2016/086.pdf

[34] S. Pinto and N. Santos, "Demystifying ARM TrustZone: A Comprehensive Survey", ACM Computuing Survey, vol. 51, January 2019, DOI 10.1145/3291047

[35] K. Eldefrawy, N. Rattanavipanon, and G. Tsudik, "HYDRA: Hybrid Design for Remote Attestation (Using a Formally Verified Microkernel)", 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Boston (MA, USA), July 2017, p. 99–110, DOI 10.1145/3098243.3098261

[36] K. Eldefrawy, A. Francillon, D. Perito, and G. Tsudik, "Smart: secure and minimal architecture for (establishing dynamic) root of trust", NDSS 2012, 19th Annual Network and Distributed System Security Symposium, San Diego (CA, USA), February 2012, pp. 1–15. https://www.eurecom.fr/fr/publication/3536/download/rs-publi-3536.pdf

[37] L. Jäger, R. Petri, and A. Fuchs, "Rolling DICE: Lightweight Remote Attestation for COTS IoT Hardware", 12th ACM International Conference on Availability, Reliability and Security, Reggio Calabria (Italy), August 2017, DOI 10.1145/3098954.3103165

[38] S. W. Smith, "Trusted computing platforms: design and applications", Springer, 2013, ISBN: 978-0-387-23917-0

[39] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song, "Keystone: An Open Framework for Architecting Trusted Execution Environments", 15th ACM European Conference on Computer Systems, Heraklion (Greece), April 2020, pp. 1–6, DOI 10.1145/3342195.3387532

[40] A. Waterman, Y. Lee, R. Avizienis, D. A. Patterson, and K. Asanovic, "The RISC-V instruction set manual volume II: Privileged architecture version 1.7", EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-49, 2015. https://riscv.org/wp-content/uploads/2019/08/riscv-privileged-20190608-1.pdf

[41] TCG, "Measurement and Attestation RootS (MARS)", https://trustedcomputinggroup.org/work-groups/mars/

[42] TCG, "Trusted Platform Module Library Part 1: Architecture", November 2019, https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part1_Architecture_pub.pdf