MATCH-IN: Mutual Attestation for Trusted Collaboration in Heterogeneous IoT Networks

Silvia Sisinni
Politecnico di Torino
DAUIN
Torino, Italy
silvia.sisinni@polito.it

Diana Gratiela Berbecaru
Politecnico di Torino
DAUIN
Torino, Italy
diana.berbecaru@polito.it

Valerio Donnini
Politecnico di Torino
DAUIN
Torino, Italy
valerio.donnini@polito.it

Antonio Lioy
Politecnico di Torino
DAUIN
Torino, Italy
antonio.lioy@polito.it

***Abstract*—As the Internet of Things (IoT) continues to evolve, ensuring the security and trustworthiness of devices within heterogeneous IoT networks becomes of paramount importance. This paper presents MATCH-IN (Mutual Attestation for Trusted Collaboration in Heterogeneous IoT Networks), a novel approach to establish trusted connections based on mutual attestation between IoT devices that dynamically join a network. Drawing inspiration from the Trusted Computing Group's "Device Identifier Composition Engine" specification, MATCH-IN introduces a comprehensive scheme for device mutual attestation. The proposed schema enhances the security posture of unstructured IoT networks by enabling devices to mutually attest their identities and configurations, without the need for a centralized verifier for checking the trustworthiness of devices, while these operate in the field. Through a detailed exploration of the DICE specification, this paper provides insights into the integration of MATCH-IN within the context of diverse IoT environments. Our approach aims to foster trusted collaboration among heterogeneous IoT devices, laying the foundation for enhanced security and reliability in the rapidly expanding IoT landscape.**

***Index Terms*—IoT, DICE, Dynamic SWARMS Attestation, Remote Attestation, Hybrid Attestation, Mutual Attestation**

I. Introduction

THE exponential growth of *Internet of Things* (IoT) devices has introduced us in an era of unprecedented connectivity, reshaping the way individuals interact with the world around them. Every aspect of our society is intertwined with IoT networks, which have become integral to the deployment of "smart" applications, offering users enhanced efficiency, automation, and convenience. The rapid advancement of wireless communication systems, along with new sensor technologies, has enabled collaboration among heterogeneous IoT devices through diverse network domains. However, this proliferation of interconnected devices has also given rise to significant security challenges. The inherent diversity and scale of IoT ecosystems create an intricate web of vulnerabilities, making them susceptible to malicious exploits and unauthorized access.

Cybersecurity of IoT ecosystems is a critical challenge for the architectural design and the operational methodologies of such systems. Traditional security mechanisms and protocols are generally not applicable to IoT devices, which are characterized by limited amounts of resources in terms of memory, power and computational capacity. This determines the need for research and industry community to develop new technologies, methodologies and standards able to address IoT security threats and guarantee an high level of integrity, privacy and reliability in IoT networks. Many innovative technologies have been proposed to solve reliability and privacy issues of IoT networks, such as Blockchains, Smart Contracts, Artificial Intelligence (AI), Federated Learning. However, the guarantee of security and trust of "smart" environments is contingent upon the verification of the integrity of the hardware and software components of all the devices that compose the IoT ecosystem.

Recognizing the critical need to verify the trustworthiness of IoT networks, our work introduces MATCH-IN (Mutual Attestation for Trusted Collaboration in Heterogeneous IoT Networks), a novel approach to establishing mutual attestation among devices within heterogeneous IoT networks. Existing literature has seen commendable efforts in proposing attestation schemas for IoT networks, particularly in the domain of SWARM (collective) attestation [1]–[4]. While these schemes address the challenges of performing an efficient collective attestation of IoT devices, they are not suitable to be applied in the context of highly dynamic and unstructured IoT networks. In such environments, unknown transient devices may join the network seeking collaboration or access to smart services. It is imperative to ensure the trustworthiness of these devices before granting them entry into the network. In our opinion, the problem of ensuring the trustworthiness of highly dynamic and unstructured IoT networks is still a challenge opened to the scientific community.

MATCH-IN proposes a mutual attestation scheme that requires minimal hardware features on IoT devices, and is rooted in the Trusted Computing Group's (TCG) "Device Identifier Composition Engine" (DICE) specification. By leveraging

This work has received funding from the SPIRS (Secure Platform for ICT Systems Rooted at the Silicon Manufacturing Process) project with Grant Agreement No. 952622 under the European Union's Horizon 2020 research and innovation programme. This work was also partially supported by the project SERICS (PE00000014) under the NRRP MUR program funded by the European Union - NextGenerationEU. Dr. Diana Gratiela Berbecaru carried out her work within the Ministerial Decree no. 1062/2021 and received funding from the FSE REACT-EU - PON Ricerca e Innovazione 2014-2020.

DICE, MATCH-IN aims to instill a heightened security posture in highly dynamic IoT networks, ensuring that all devices, even those that temporarily transit the network, undergo a rigorous attestation process before joining a “smart” environment.

This paper unfolds with an exploration of the most significant works related to IoT attestation (II), introduces the main concepts related to the TCG DICE and Open DICE specifications (III), presents MATCH-IN as a viable solution to address the security concerns in highly dynamic IoT ecosystems (IV), and concludes by summarizing the contribution of this work (V).

II. Related Works

In recent years, IoT device attestation is becoming a particularly popular and crucial challenge for cybersecurity researchers due to the exponential growth of infrastructures that base their operation on IoT networks. The first works date back to around the 2000s, when the first release of the TPM chip began to be almost ubiquitous on desktop and server platforms enabling remote attestation on this category of devices, so researchers began to look for similar solutions suitable for very simple devices that constituted the so-called “wireless sensors networks”. Since the TPM was too complex to be inserted into very small and low-resource devices, researchers initially oriented their efforts towards *Software-based* solutions that achieved remote attestation by relying on the knowledge of the exact computational capabilities available to such devices [5], [6]. However, between 2008 and 2009 it was proven that those solutions were vulnerable and could be easily circumvented, so researchers sought to define the minimal set of hardware requirements that an IoT device had to support to enable remote attestation. It was understood that the minimum hardware support consisted of a Read Only Memory (ROM) to store the attestation protocol and the attestation key, and hardware protection of the private part of the attestation key in order to ensure that this can only be accessed by the attestation protocol [7], [8]. These are the minimum requirements to ensure that only the device’s trusted code, i.e. the attestation protocol stored in ROM, has signed the attestation report, and constitute what is called *Hybrid-based* attestation, on which most IoT attestation protocols are still based today.

Until 2015, researchers had focused on enabling remote attestation for a single device, relying on a classic attestation scheme, where a single verifier is responsible for attestation of each device in the monitored network. However, this model proves to be inefficient for networks made up of a high number of IoT devices, as happens for example in Industrial IoT, Smart Cities and Smart Transportation installations. In 2015, SEDA (Scalable Embedded Device Attestation) [1] was published, the first paper that presented a *SWARM* attestation scheme with the aim of efficiently attesting a large number of IoT devices. SEDA assumes that all devices in the network have the requirements of the Hybrid attestation, that the configuration of the IoT network is static, both in terms of the number of devices and their positioning, and that each device can only talk to its neighbors. To attest the integrity status of the network, the SEDA protocol provides a single verifier that sends an attestation request to a randomly chosen device in the network. The chosen device becomes the root of a Spanning Tree calculated over the entire network of IoT devices and is responsible for propagating the attestation request to all its children until such request reaches the leaves of the tree. Each leaf calculates its own attestation report and sends it to the parent device; this aggregates the attestation reports of all the children with its own report and in turn sends the aggregate result to its parent. This procedure is repeated until the root, which finally sends the overall response to the verifier. From the attestation response, the verifier can establish whether the entire network of devices is trusted or not, without being able to identify any devices that may have been compromised, in the case of an untrusted response. SEDA influenced many subsequent works [2], which improved the scheme for detecting physical attacks, identifying the compromised devices, and making the aggregation of results more effective using Merkle Hash Trees. However these schemes, being based on the calculation of a Spanning Tree, are not flexible, so they are not applicable in contexts in which IoT devices can move in space, as the Spanning Tree would have to continuously reconfigure itself and the absence of cycles in the graph could not be guaranteed.

The problem of attesting networks of heterogeneous and highly dynamic IoT devices was addressed in 2018 by two works, SALAD [3] and PADS [4], which proposed a different approach with respect to previous ones, based on the gossip protocol and a mechanism of distributed consensus. The *Dynamic SWARMS* model takes into account a highly dynamic and unstructured network of IoT devices. During its movement in space, a device can encounter other devices that fall within its communication range and perform a mutual attestation with them. Each device is able to establish the level of reliability of other devices by comparing the attestation report with the expected measurement, stored in a trusted component. If devices are trusted, they agree to exchange their historical attestation results relating to all other devices in the network. The final view of the integrity status of the network is built through the concept of consensus, in which the level of trustworthiness of each device is the result of the logical AND or XOR operation on the evaluations possessed by the two devices before sharing their information. This protocol is suitable for attesting highly dynamic networks topologies, as it allows device movement during attestation; however, its complexity makes it not suitable for devices with very few computational resources. Moreover, it still has a strong limitation in terms of network dynamism as it is not applicable to scenarios in which a new device arrives and dynamically joins the network.

The problem of defining an attestation protocol for highly dynamic IoT networks, in which new devices join the network and need to be attested before being accepted as trusted components of the infrastructure, still constitutes an open challenge for the scientific community. This paper fits into this research area, proposing a model based on DICE concepts, embedded in firmware and IoT applications, to enable mutual attestation of IoT devices in highly dynamic environments.

III. Background

In the ever-expanding and evolving landscape of IoT networks, establishing a solid root of trust in IoT devices is essential to ensuring the integrity of connected devices and the reliability of the IoT ecosystem. Remote attestation is a challenge-response protocol that allows a remote entity, the so-called verifier, to state the integrity status of the hardware and the software of a platform. In order to enable remote attestation, it is imperative that the device is provisioned with a set of system elements that are reliable by design and which cannot be compromised at runtime by an attacker. However, the intrinsic hardware limitations of IoT devices pose a significant challenge. Traditional solutions, such as the Trusted Platform Module (TPM), which serves as a hardware root of trust for desktop and server platforms, are often impractical for resource-constrained IoT devices. In response to this challenge, significant efforts have emerged to define new specifications that enable roots of trust on highly resource-constrained devices. Two significant proposals in this research area are represented by the TCG's "Device Identifier Composition Engine" (DICE) specification and the Open DICE initiative. These specifications provide a framework for constructing unique and immutable device identifiers that serve as a foundation for remote attestation. By addressing the hardware limitations of IoT devices and offering a standardized approach, TCG DICE and Open DICE contribute significantly to enhancing the security landscape of the IoT, opening doors to trustworthy and interoperable device identification and attestation in a dynamic and evolving scenario.

A. TCG DICE

The TCG DICE specification defines cryptographic identities for a device's hardware and firmware, useful for attesting the instantiation of a Trusted Computing Base (TCB) during a device's boot sequence. At the heart of DICE is the construction of a *Compound Device Identifier* (CDI), encapsulating the amalgamation of hardware and software components crucial to the device's security [9]. DICE defines a layered architecture, which relies on a hardware Root of Trust (HRoT) as the foundation for a multi-layered TCB. This architecture facilitates secure transitions between layers, each marked by the creation and secure transmission of CDI values [10]. Fig. 1 schematizes the CDI generation process for the different TCB layers in a platform, starting from a *Unique Device Secret* (UDS).

The HRoT must adhere to specific requirements, affordable for IoT devices. These include the provisioning of the UDS, access limitations to TCB layers' secrets (e.g. CDIs), and assertions of trustworthiness, typically provided through certificates, by hardware manufacturers or vendors. The DICE

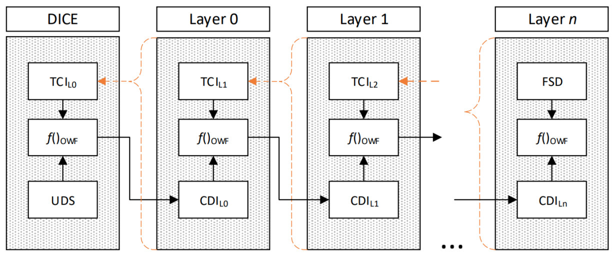


Fig. 1. TCB layering architecture [10].

specification also defines the derivation of cryptographic keys from the CDI of each TCB layer, in particular three types of asymmetric keys: *Embedded Certificate Authority* (ECA) keys, used by TCB layers to issue certificates; *Attestation Keys*, used to sign attestation evidence; *Identity Keys*, used for signing authentication challenges. Cryptographic keys, derived according to TCG DICE, can be endorsed with a certification hierarchy that uses a *Public Key Infrastructure* (PKI) for device provenance. This chain of trust, established outside the device, can be extended to the various TCB layers by means of the Embedded Certificate Authority (ECA) concept. Fig. 2 shows an example certificate hierarchy consisting of a Root CA that endorses an intermediate CA that issues an end-entity certificate.

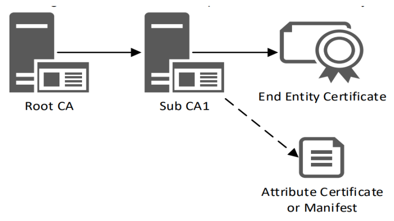


Fig. 2. Certificate hierarchy [10].

IV. MATCH-IN: System Model

This work proposes MATCH-IN, a software model based on the DICE specification to enable trustworthy interaction and implicit attestation between unknown devices on highly dynamic and untrusted networks. MATCH-IN differs from the Dynamic SWARMS models presented in literature [3], [4] as it does not provide for the presence of a centralized verifier responsible for monitoring the state of integrity of the entire network during runtime, but presents a completely decentralized model for verifying the state of reliability of the devices. In this model, the devices, before starting an application interaction, mutually attest their integrity status and, seamlessly, are able to recognize and isolate compromised devices from the "smart" network. This model can be used when it is not possible to predict a priori the devices that will be part of the network and which, at a given moment, will enter the IoT ecosystem to contribute to the provision of a service or make use of a service by connecting to other devices, as happens for example in Smart Cities, Smart Healthcare, Smart Transportation use cases.

A. Requirements

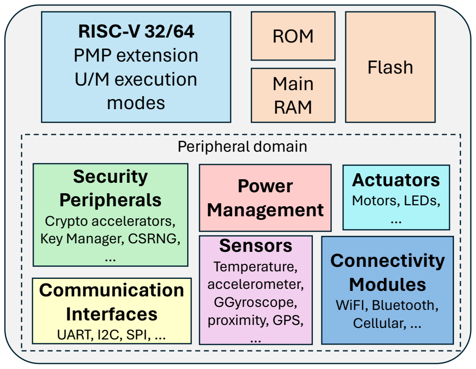


Fig. 3. Typical hardware modules embedded on an IoT device with RISC-V core.

MATCH-IN is based on some hardware and software requirements that the device has to possess and on certificate provisioning procedures that the device manufacturer and the application provider has to implement. In particular, it is assumed that the device is equipped with minimum hardware requirements: a Unique Device Secret (UDS) that uniquely identify the device, hardware mechanisms to protect specific memory regions and a Boot ROM enforcing secure boot of the first stage firmware.

Figure 3 shows a high-level schema of the hardware components present in typical IoT devices. We have taken into consideration IoT devices equipped with a RISC-V processor supporting: two execution modes, *Machine mode* (M-mode), corresponding to the highest privilege, *User mode* (U-mode), corresponding to the least privilege; the *Physical Memory Protection* (PMP) security extension, used by code running in M-mode to limit access permissions to specific memory areas. The device is equipped with ROM memory containing the first code executed by the device at power-on and implementing secure boot and chip configuration. Moreover, the device is provided with flash memory as a non-volatile storage medium to store the firmware provided by the device manufacturer and the device owner. The device could also be equipped with hardware modules to accelerate cryptographic primitives (e.g. AES, HMAC, KMAC, cryptographically secure random number generator), sensors, actuators and connectivity modules, depending on the specific applications and use cases in which the IoT device is designed to be used.

B. Bootflow and Cryptographic Identity Generation

The initial bootstrapping of the device starts from the Boot ROM code. This is defined in the hardware design of the device and remains immutable throughout all its life cycle; it contains the minimum set of instructions needed to initialize the hardware peripherals, force the secure boot of the next boot stage.

If the signature verification of the next firmware payload is successful, the Boot ROM code jumps into the next boot stage, which implements the so-called “DICE Core”, described in the DICE hardware specification [9]. This code, unlike the Boot ROM, can be updated after the device has been manufactured, provided that it has been signed by the manufacturer with a key corresponding to the public key embedded in the Boot ROM. The “DICE Core” firmware first forces secure boot of the next boot stage, corresponding to the device owner’s firmware. In the case of low-end IoT devices, the device owner’s code is the application implementing the use case in which the device will be used; in the case of medium- and high-end devices, the next stage is the bootloader of a lightweight operating system capable to run multiple applications. In order for the “DICE Core” to enforce secure boot, the device owner’s code must be associated with a manifest, stored in flash memory at a specific address and containing the signature of the firmware payload and the public key with which the signature will be verified. If the verification is successful the bootflow continues, otherwise the device fails to boot. The firmware manifest is protected from malicious modifications, which can occur at runtime; the “DICE Core”, before jumping to the device owner’s code, uses the PMP primitive to eliminate write access permissions from the memory area containing the manifest, until the next power-on of the device.

After enforcing the secure boot, the “DICE Core” derives the *Device Identity Key* (DIK) by mixing information that identifies the device at the hardware level (2): the “Unique Device Secret” (UDS), a statistically unique secret value per device, provided at manufacturing time and stored in a “One Time Programmable” (OTP) memory, or derived from a hardware module implementing a “Physical Unclonable Function” (PUF); the identity of the Boot ROM and “DICE Core” code, represented by the digest calculated on the respective memory regions; we call this identity *ROM Code Identifier* (RCI) (1).

$$RCI = Hash(Boot\ ROM\ ||\ DICE\ Core) \tag{1}$$
$$DIK = f_{KeyGen}(Hmac(UDS,\ RCI)) \tag{2}$$

The DIK plays the role of *Embedded Certificate Authority* (ECA) and can only be used to sign the certificate identifying the device owner’s firmware. This allows to create an attestation chain based on the authenticity of the hardware and the first two boot stages, provided by the device manufacturer.

After generating the DIK, the “DICE Core” generates the so-called *Compound Device Identifier* Level 0 (CDI_0), as it is indicated in the TCG DICE specification [10]. CDI_0 is obtained by combining the device’s hardware identity (i.e. UDS and RCI) with the owner’s firmware identity, which can be directly the application or the bootloader (3). The “DICE Core” uses the CDI_0 to derive the *Owner Identity Key* (OIK) (4).

$$CDI_0 = Hmac(UDS||RCI,\ Hash(owner's\ code)) \tag{3}$$
$$OIK = f_{KeyGen}(CDI_0) \tag{4}$$

Then, the "DICE Core" creates a certificate associated with the OIK, $Cert_{OIK}$, signs it with the private part of the DIK (DIK_{priv}) (5) and stores it in an area of flash memory accessible to all boot phases, including application layer.

$$Sign_{DIK_{priv}}(Cert_{OIK}) \tag{5}$$

According to the TCG DICE specification [11], $Cert_{OIK}$ has extensions, containing the digest of the owner's firmware and the hash algorithm with which it was computed. These extensions will be used during firmware attestation. Before jumping into the device owner's firmware, the "DICE Core" copies the CDI_0 in a memory area where the next stage firmware can read it, and appropriately configures the PMP registers to: remove read permissions from the UDS, wipe RAM and CPU memory of any reference to DIK_{priv}, and remove execution permissions to the memory area containing the "DICE Core", in order to prevent subsequent software layers from being able to regenerate and use the DIK_{priv}.

The following description takes into consideration the case in which the device owner's firmware is the bootloader and the kernel; the simplest case, in which the application runs directly on the hardware, can be easily derived from this. The bootloader, in turns, can enforce the secure boot of the kernel, if this has been provided with a manifest containing the kernel signature and the public key with which it must be verified, proceeding as for the secure boot of the previous stage. Then, the bootloader regenerates the OIK starting from CDI_0 (4), measures the kernel image and generates CDI Layer 1 (CDI_1) from kernel digest and CDI_0 (6), generates the kernel level ECA key starting from CDI_1 (7), creates a certificate for the kernel ECA, containing the kernel digest and the hash algorithm as extensions, signs it with the private part of the OIK (OIK_{priv}) (8), copies $Cert_{Kernel\ ECA}$ and the CDI_1 into a memory area accessible to the kernel, and finally boots the kernel.

$$CDI_1 = Hmac(CDI_0,\ Hash(Kernel\ image)) \tag{6}$$
$$Kernel\ ECA = f_{KeyGen}(CDI_1) \tag{7}$$
$$Sign_{OIK_{priv}}(Cert_{Kernel\ ECA}) \tag{8}$$

When the kernel is booted, it regenerates its ECA key starting from CDI_1 (7).

When an application is loaded, the kernel carries out a procedure similar to what is described for the previous layers. First of all, it computes the digest of the application binary and uses it, together with CDI_1, to generate CDI Layer 2 (CDI_2) (9); generates a *Local Device Identity Key* (LDevID), using the CDI_2 as seed (10); creates a certificate for the LDevID Key, containing the application digest and hash algorithm as extensions, and signs it with the kernel ECA key (11); finally, it passes the CDI_2 and the $Cert_{LDevID}$ as application parameters.

$$CDI_2 = Hmac(CDI_1,\ Hash(Application\ binary)) \tag{9}$$
$$LDevID = f_{KeyGen}(CDI_2) \tag{10}$$
$$Sign_{Kernel\ ECA_{priv}}(Cert_{LDevID}) \tag{11}$$

The application can use CDI_2 to regenerate its LDevID key (10), which represents the cryptographic identity of the application running on a specific device.

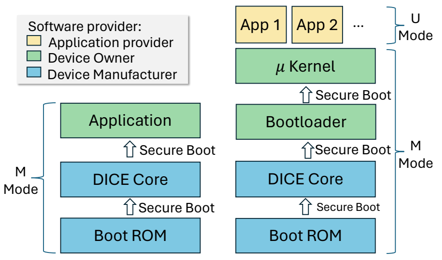


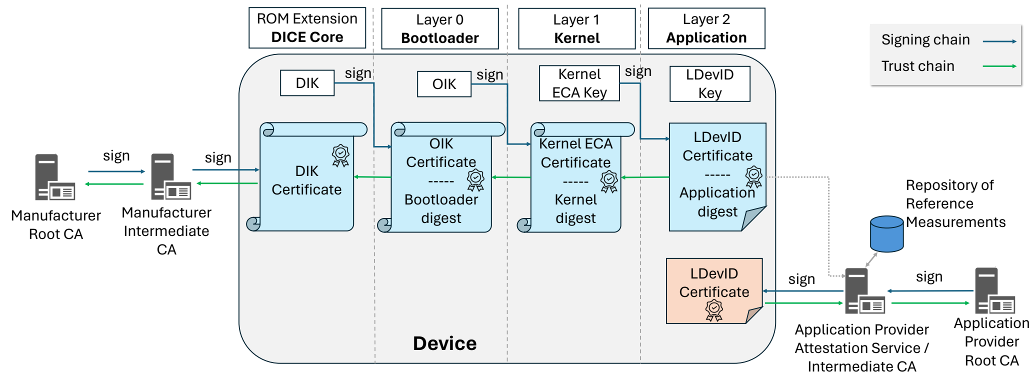
Fig. 4. Bootflow scheme for low-end devices (left) and medium- high-end devices (right).

Figure 4 represents at a high level the bootflow stages of an IoT device, both in case of low-end devices and in case of medium and high-end devices.

C. Certificate Provisioning from External PKIs

The previous section describes the generation of certificates based on Embedded CAs internal to the IoT device, starting from the DIK. This key can be provisioned with a certificate signed by a *Manufacturer Intermediate Certificate Authority* (CA), which can be verified using a stable Public Key Infrastructure (PKI), as shown in Fig. 5. This certificate represents the Root of Trust on which the device attestation is based, therefore it guarantees the trustworthiness of the device's hardware: it ensures that the device has been properly manufactured and tested, and that the device provides the minimum system elements to allow remote attestation. Alternatively, the "DICE Core" can generate a self-signed certificate for the DIK, and the endorsement of the DIK must take place by comparing the public key against a trustworthy device registry.

The manufacturer-based certificate chain represented in Fig. 5, can be used by the application provider as attestation evidence for verifying the trustworthiness of the application running on a specific device. In particular, the application provider can provide a "attestation provisioning service" with the aim of verifying the certificate chain of the application and issuing a new certificate associated with the LDevID key of the application, through an Application Intermediate CA, as represented by the pink LDevID certificate in Fig. 5. The application provider's "attestation provisioning service" performs the following steps: acquires the attestation certificate chain based on the device manufacturer PKI, up to the LDevID certificate; checks the validity of the entire certificate chain by checking the signatures; checks the authenticity of the device hardware and firmware by comparing the measurements contained in the certificates with reference measurements contained in a trustworthy repository; if the previous checks are successful, requests the Application Provider Intermediate CA to issue a certificate, and sends it to the IoT device. In this way, the application provider's "attestation provisioning



ROM Extension
DICE Core
Layer 0
Bootloader
Layer 1
Kernel
Layer 2
Application
Signing chain
Trust chain
DIK
sign
OIK
sign
Kernel ECA Key
sign
LDevID Key
DIK Certificate
OIK Certificate ----- Bootloader digest
Kernel ECA Certificate ----- Kernel digest
LDevID Certificate ----- Application digest
sign
sign
Manufacturer Root CA
Manufacturer Intermediate CA
Repository of Reference Measurements
LDevID Certificate
sign
sign
Application Provider Attestation Service / Intermediate CA
Application Provider Root CA
Device

Fig. 5. Layered certificates example.

service" performs a remote attestation of the device hardware and firmware and, if the device is evaluated trustworthy, issues an application-level certificate which establishes the overall trustworthiness of the IoT application running on a specific device.

D. *Implicit attestation among IoT devices*

When two IoT devices enter their communication range and need to establish a communication channel to exchange data, they can open a mutually authenticated TLS channel by relying on the respective LDevID keys associated with the IoT application, and on the LDevID certificate issued by the application provider according to the procedure described in section IV-C. In this way, the establishment of the TLS connection constitutes implicit evidence of the trustworthiness of the application and of the device on which it is running. This is because, if the application has been tampered with, and/or if any of its underlying software and hardware components have been compromised, the LDevID key generated by the application will differ from the one certified by the application provider, and the communication channel cannot be opened.

This procedure allows trustworthy collaborations between unknown IoT devices, as long as they have installed compliant software, and at the same time it seamlessly isolates IoT devices that are compromised.

V. Conclusions

The exponential growth of IoT devices has brought unprecedented connectivity and convenience but also raised significant security challenges. This paper introduces MATCH-IN, a schema for mutual attestation which relies on TCG DICE specification for ensuring robust security in heterogeneous IoT networks, with minimal hardware requirements. While existing attestation schemes address the problem of collective attestation in IoT networks made up of a fixed number of devices, MATCH-IN proposes a solution to ensure trustworthy collaborations in highly dynamic IoT environments. Leveraging the DICE specification, MATCH-IN ensures a heightened security posture, subjecting all devices to an attestation procedure before allowing them to communicate.

References

[1] N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann, "SEDA: Scalable Embedded Device Attestation," in *22nd ACM SIGSAC Conference on Computer and Communications Security*, Denver (CO, USA), Oct. 12–16 2015, p. 964–975. [Online]. Available: https://doi.org/10.1145/2810103.2813670

[2] M. Ambrosin, M. Conti, A. Ibrahim, G. Neven, A.-R. Sadeghi, and M. Schunter, "SANA: Secure and Scalable Aggregate Network Attestation," in *ACM SIGSAC Conference on Computer and Communications Security*, Vienna (Austria), Oct. 24–28 2016, p. 731–742. [Online]. Available: https://doi.org/10.1145/2976749.2978335

[3] F. Kohnhäuser, N. Büscher, and S. Katzenbeisser, "Salad: Secure and lightweight attestation of highly dynamic and disruptive networks," in *Asia Conference on Computer and Communications Security*, Incheon (Republic of Korea), June4 2018, p. 329–342. [Online]. Available: https://doi.org/10.1145/3196494.3196544

[4] M. Ambrosin, M. Conti, R. Lazzaretti, M. Rabbani, and S. Ranise, "PADS: Practical Attestation for Highly Dynamic Swarm Topologies," in *International Workshop on Secure Internet of Things (SIoT)*, Sept. 6 2018, p. 18–27.

[5] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "SWATT: softWare-based attestation for embedded devices," in *IEEE Symposium on Security and Privacy*, Berkeley (CA, USA), May 9–12, 2004, pp. 272–282. [Online]. Available: https://doi.org/10.1109/SECPRI.2004.1301329

[6] 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," in *12th ACM Symposium on Operating Systems Principles*, Brighton (UK), Oct. 23–26, 2005, p. 1–16. [Online]. Available: https://doi.org/10.1145/1095810.1095812

[7] K. Eldefrawy, A. Francillon, D. Perito, and G. Tsudik, "SMART: Secure and Minimal Architecture for (Establishing a Dynamic) Root of Trust," in *19th Annual Network and Distributed System Security Symposium*, ISOC, Ed., San Diego (CA, USA), Feb. 5–8, 2012.

[8] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan, "TrustLite: a security architecture for tiny embedded devices," in *9th European Conference on Computer Systems*, Amsterdam (Netherlands), Apr. 14–16, 2014. [Online]. Available: https://doi.org/10.1145/2592798.2592824

[9] Trusted Computing Group (TCG), "Hardware Requirements for a Device Identifier Composition Engine," https://trustedcomputinggroup.org/wp-content/uploads/Hardware-Requirements-for-Device-Identifier-Composition-Engine-r78_For-Publication.pdf, Trusted Computing Group, Tech. Rep. Level 00 Revision 78, Mar.22 2018.

[10] ——, "DICE Layering Architecture," https://trustedcomputinggroup.org/wp-content/uploads/DICE-Layering-Architecture-r19_pub.pdf, Trusted Computing Group, Tech. Rep. Version 1.0 Revision 0.19, July 23 2020.

[11] ——, "DICE Attestation Architecture," https://trustedcomputinggroup.org/wp-content/uploads/DICE-Attestation-Architecture-Version-1.1-Revision-18_pub.pdf, Trusted Computing Group, Tech. Rep. Version 1.1 Revision 0.18, Jan.6 2024.