IEEE ICBC'24 1570977431

DIMSIM – **D**evice **I**ntegrity **M**onitoring through **iSIM** Applets and Distributed Ledger Technology

***Abstract*—In the context of industrial environment, devices, such as robots and drones, are open and easy target for malicious activities such device tampering (e.g., hardware and software changes). The problem becomes even worse in a multi-stakeholder environment where multiple players contribute to an ecosystem. In such scenarios, that is, when devices are deployed in remote settings, ensuring device integrity so that all stakeholders can trust them is challenging. Existing methods, often depend on additional hardware like the Trusted Platform Module (TPM) which may not be implicitly provided by all vendors. In this study, we introduce a distributed ledger technology oriented architecture to monitor the remote devices' integrity using eUICC technology, a feature commonly found in industrial devices for connectivity. We propose that using eUICC applets, devices' integrity can be monitored and managed without installing any additional hardware.**

I. Introduction

Future industrial systems, such as Industry 4.0 [1] and Industry 5.0 [2], are envisioned as multi-stakeholder environments— a multi-actor and open ecosystem [3] — in which assets and services may not necessarily come from a limited number of providers. These systems provide opportunities for a wide range of vendors, regardless of their business size, to contribute to the ecosystem. The primary concern in such systems is trust among the stakeholders within the ecosystem [4], [5]. We believe that such open systems can only be enabled *if and only if* the systems are inherently *trustable* and *accountable*.

This assertion is grounded in the following primary reasons: The first challenge related to open systems [3] is their ability to easily interoperate with devices from a wide range of providers and form end-to-end heterogeneous systems. In such heterogeneous systems various device providers will be involved in an end-to-end service. Therefore, each party depends on other parties for the smooth operations and quality of service delivered. Hence, malfunctioning of a single device will impact the performance of an overall service. The challenge is not limited to quality but also for security reasons; a corrupt or poorly secured device can introduce dangers to the full system [6]. This necessitates robust monitoring and reporting mechanisms that allow complete transparency among actors, thereby establishing trust.

The second challenge revolves around establishing accountability for devices in large-scale deployments featuring multiple participants in the ecosystem. In industrial contexts, ensuring the accountability and reliability of remote devices poses a significant challenge. When devices operate remotely, their performance may deviate from the intended programming. This discrepancy could stem from various factors, including malicious interference by external entities altering device software to compromise the environment or derive personal gains. Thus, devices integrated into industrial environments must inherently embed mechanisms ensuring not only the integrity of their software and firmware but also providing unequivocal assurances regarding the accuracy and accountability of the data they generate. A robust system of accountability is crucial to uphold the reliability and trustworthiness of the entire ecosystem.

Hence, future generation of industrial systems must be both open and accountable, inherently ensuring trust. Data generated by these devices and transactions between stakeholders must be accurate, trustworthy and reliable for all the participants. Additionally, these systems should guarantee the proper execution of devices' programmed functions, while also detecting and taking appropriate actions for any malicious behaviors. The entire ecosystem should operate autonomously, functioning in a zero-touch manner.

To this end, our work introduces an end-to-end device integrity monitoring system designed for a multivendor environment. The system monitors device integrity and enables trust, without the need for additional hardware: DIMSIM (Device Integrity Monitoring with SIM Applets) (Fig.1). We leverage eUICC technology to maintain device integrity. eUICC is not an additional burden on the device and is a standard to provide connectivity to remote devices [7].

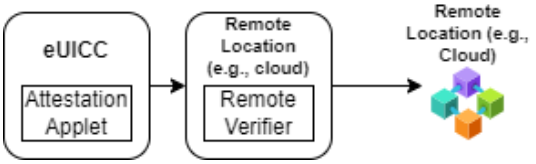

Fig. 1. DIMSIM – Modular Architecture of DIMSIM consists of 1) an Attestation Applet, 2) a Remote Verifier and c) a Permissioned Distributed Ledger

DIMSIM is a modular system leveraging the combination of the eUICC (eSIM or iSIM) technology at the device level, and Distributed Ledger technology at the cloud level (Fig. 1). We exploit eUICC in combination with distributed ledger technology to provide all the stakeholders a transparent view of the device software and firmware. We use a specific type of distributed ledgers – "Permissioned Distributed Ledger (PDL)" due to that the fact they have already been discussed widely in industrial applications, for example, in operations and control

networks [4], automated monitoring [8] and network devices' sharing [9].

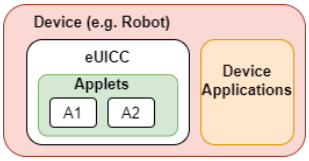

Fig. 2. Simpler eUICC Architecture highlighting Applets (Secure Elements). The eUICC Applets are inside a secure area within the eUICC, and can only be accessible by the mobile network provider (MNO). Here, A1 and A2 represent applets installed by the MNO.

II. Related Work

In this work, we introduce a system designed to facilitate a multivendor environment. Our objective is to enable stakeholders to precisely assess the integrity of each individual device. We achieve this with the combination of eUICC and distributed ledger technologies.

Several systems to monitor device integrity are proposed such as I^3 [10] and Tripwire [11]. Such proposals enable administrators to verify device integrity but do not address the concern of malicious device deliberately tampering with the device.

An alternate to Trusted Platform Module (TPM) within the domain of eUICC (embedded Universal Integrated Circuit Card) is proposed by Chakraborty et al. [12] in *SimTPM*. They proposed incorporating TPM functionality in eUICC. SimTPM utilizes the eUICC's non-volatile memory to store device measurements. Unlike us, they use TPM commands for integrity monitoring and do not perform real-time monitoring. Neither simTPM automatically reports device changes to a remote location (e.g., device lessee).

In the context of mobile devices, Raj et al. introduced *firmwareTPM (fTPM)* [13], a solution specifically tailored for mobile devices that relies on ARM TrustZone for its operation.

Petroni et al.'s *Copilot* [14] aligns closely with our proposal, although it necessitates the use of a PCI add-in card. This PCI card plays a critical role in detecting malicious modifications to the kernel.

Our solution surpasses the limitations of TPM functionalities and eliminates the dependency on additional hardware. Notably, it's important to mention that eUICC is already a standard component in devices requiring connectivity. Our framework, DIMSIM, represents an end-to-end solution that continuously monitors device integrity and reports any anomalies detected to the stakeholders.

Furthermore, DIMSIM offers a transparent view to all stakeholders and ability to stop the device, if an anomaly is detected. Such functionalities are crucial for ensuring the viability of future industrial applications.

III. Architecture

Three different types of entities interact with this system a) **Solution Provider** – an entity which forms solutions by combining different services (e.g., connectivity) and devices (e.g., robots), b) **Device Vendor** – device provider who leases their devices to solution providers, and c) **Service Provider** - entities which provides services such as connectivity and robot controller.

In addition to the actors, there are following main components for our architecture:

- **Assets** – For example, devices and services available on a platform such as a marketplace [15]. The devices have connectivity through eUICC and secure elements are enabled.
- **Attestation Applet** – our proposed novel secure element which monitors the device integrity. The Applet is controlled by the solution provider. The attestation applet ensures device integrity by continuous monitoring, reporting anomalies to the remote verifier.
- **Remote Verifier** – an immutable and untemperable entity managed by all the stakeholders It maintains and manages immutable database and record of all good-known values of devices software and firmware. However, the Remote Verifier cannot update the records themselves. They must take consensus from all the stakeholders to update the records. The remote verifier may be running on a platform such as edge cloud.
- **Permissioned Distributed Ledger (PDL)** – a distributed ledger managed by all the stakeholders within the ecosystem. We use a permissioned type of distributed ledger, which is managed by the consortium of stakeholders within the ecosystem. The transactions and contracts among the participants of the ecosystem are recorded in the PDL, such as service level agreements. The ledger is consensus agnostic.

Our system is formed by participants running a permissioned distributed ledger (PDL) that includes agreements between the actors within the ecosystem. Assets (e.g., services, devices) and their associated contracts (e.g., service level agreements (SLAs)) are deployed as smart contracts on the PDL. When a solution provider forms a solution by leasing devices, service level agreements between the device vendors and solution provider are established. When a customer wishes to purchase a solution, they generate a request for resource(s). The Solution Provider executes corresponding SLA and the service (if available) is allocated to the customer. As soon as the SLAs are established, that is, appropriate smart contracts are executed, the service is started. The details of the resource provisioning are out-of-scope of this work and interested readers can refer to [5] for details.

Our goal is to ensure that the allocated resources (e.g., devices) perform as promised in their SLAs throughout their lifecycle. They must provide accurate data for management purposes, such as billing and charging. To achieve this objective, it is imperative that the devices must not be tempered by any party in any manner. To ensure this, we implement the following steps:

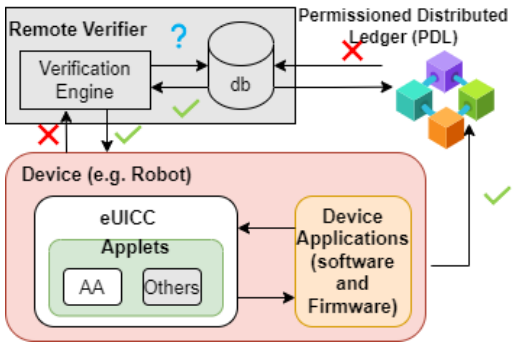

Fig. 3. Device Integrity Monitoring with eUICC applets and PDL

A. Initial Benchmark Measurements

First step is to get the record initial measurements of the device software/firmware. To obtain the initial measurements, the hash value(s) of the software and firmware are recorded in an SLA (i.e., smart contract). Note that, the hash value is the complete hash of the software or firmware code. We advocate for a trustable and auditable system; therefore, the software and firmware installed on the device must be available on a location accessible by all stakeholders (e.g., edge cloud, git repository) along with their respective hash values. To protect Intellectual Property of the software, only the hash value of the software/firmware code may be available.

Once the solution is delivered to a customer's premises, the solution provider provisions the devices (e.g., robots). Device provisioning includes installing a connectivity profile and verifying the device software and firmware. Additionally, during the device provisioning, a secure element is also installed and configured as the 'Attestation Applet' (AA) – our novel applet, which is designed to monitor device integrity. The Attestation Applet hashes the device firmware and software, send these measurements to the solution provider.

To confirm the device software and firmware are in the same state as the device vendor has promised in the contract, the solution provider compares the measurements with those provided by the device vendor with the measurements submitted by the Attestation Applet. If the measurements match, the solution provider will send a confirmation receipt to the AA, and the measurements sent by the AA, will be recorded as initial measurements inside the applet's non-volatile memory. The AA will use these values as the benchmark values for later comparison. The process is depicted in Fig(4).

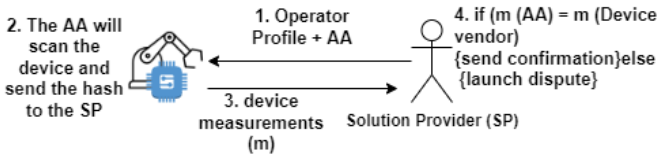

Fig. 4. Attestation Applet provisioning and Initial Measurements Process

Since the SLA, that is, the smart contract recorded in the PDL, contains the agreed-upon measurements, alternatively, the AA can send measurements directly to the PDL. However, in such a case, AA will have to wait for transaction approval, which can take longer. Additionally, PDLs are limited by the bandwidth they can handle, when a large number of devices send requests, it can lead to congestion and increased risk of transaction rejection.

B. Periodic Measurements with Attestation Applet

After the initial measurements are recorded inside the device and the solution is initiated, the Attestation Applet scans the device firmware and software periodically and matches with the values recorded in the non-volatile memory from the previous step (Fig. 3).

If the values measured by the AA match with the recorded values, AA updates its $log(t\_s, current\_hash, previous\_hash, action\_taken)$ with previous hash value similar to the current hash and action taken as $null$ (Table I) and waits for the next epoch to scan. If the values do not match, the AA updates its log and sends a message (Fig. 5) to the Remote Verifier. Optionally, if programmed, the AA can take preventive measures on its own such as stopping the device and terminate connectivity.

When the dispute message is received from the Attestation Applet, the verifier checks its own database for the records. If the 'disputed hash' matches with its records, the remote verifier notifies the AA to update its records. If they do not, the remote verifier can take further actions such as stopping the device through control messages to the eUICC.

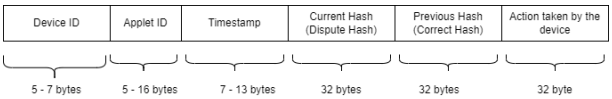

Fig. 5. Dispute data packet sent by the Attestation Applet. $CurrentHash$ is the hash value calculated by the AA and did not match with its stored and correct measurements ($PreviousHash$) in the previous epoch. $ActionTaken$ is the device's immediate action after the scan result. The possible values for Action Taken are listed in Table I .

ID	Action
0x00	*null*
0x01	*Initiate investigation*
0x02	*Restrict application or software execution*
0x03	*Isolate device*
0x04	*Contain device*
0x05	*Revoke device*
0x06	*Stop and quarantine a file*
0x07	*Request deeper investigation*

TABLE I
COMMANDS FOR ACTION TAKEN

C. Device Software/Firmware Updates

As devices are part of a solution, when a device vendor wants to update their software and/or firmware, the updates to one device can impact the performance of the complete solution, therefore solution provider must agree to the new software updates. To that end, device vendor notifies the solution provider with intended software/firmware updates and its hash value. If the solution provider agrees with the

updates, they will send a confirmation. The device vendor then executes the software update smart contracts with the updated software/firmware hash and confirmation receipt from the solution provider. The device vendor also sends a notification to remote verifier that the software/firmware is updated, and send updated hash of the software/firmware to update their records. If the solution provider does not agree with the update, they can launch a dispute and settle as per the SLA.

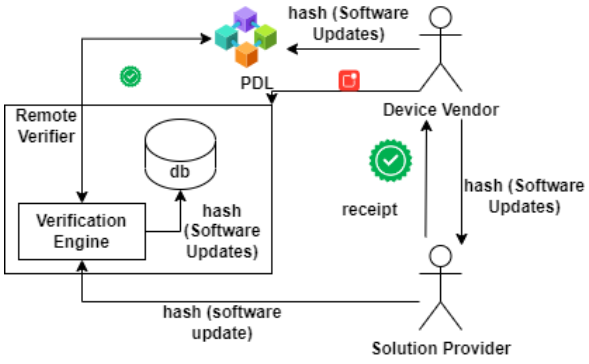

Fig. 6. Software Updates on the Remote Verifier

IV. Considerations

A. Storage

Traditional Secure Elements typically can store 4kb of application data [16]. This means that, with the given capacity, our log file can hold up to 125 hashes, each consisting of 32 bytes, at any given time. Once this limit is reached, the log file will have to be archived a remote location, for example, transferred to the Remote Verifier's immutable database.

EUICC depends on manufacturers design. New eUICC designs are still in the planning phase and the manufacturers are defining their proprietary applet sizes [17]. As per the GSMA, the memory size of an eUICC can range from several kilobytes to several megabytes, with no specific limit on the number of applets that can be installed [18]. For the Attestation Applet, the storage requirement depends on a use case and the required frequency of device integrity verification is required.

B. Response Time

In DIMSIM, the AA will notify the remote verifier for any unexpected measurements. However, once the AA identifies a corrupted file and then notifies the remote verifier, in the meantime, a compromised/corrupted device can perform malicious activities, such as spreading a virus to its nearby devices.

To that end, AA can block the device independently and before notifying the remote verifier. However, in such a setting, the device will have to be stopped for false alarms.

C. Corrupted Remote Verifier

In DIMSIM, the Attestation Applet, sends a disputed hash to the remote verifier for the verification. If the remote verifier is compromised, it can reject the claims from the AA and allow a corrupted device to continue its operations.

Recall that, the AA has direct access and authority to stop the device. The AA can be programmed in such a way to take preventive measures itself. For example, after multiple alarms of malfunctioning, the AA can stop the device and send a notification to the solution provider. In another solution example, multiple remote verifiers can be deployed, and action will be taken only after collecting consensus from all the remote verifiers. However, multiple remote verifiers may introduce additional delays to the system.

V. Evaluation Setup & Preliminary Results

Usually, eUICC supports different connectivity options to the host device. The indirect communication via a modem, based on ISO7816 (T=0 protocol). The direct communication is based on protocols such as I2C or SPI. Our experimental setup is based on a Raspberry pi Model 4B, 4G LTE base Hat, Quectel EC25-E 4g/LTE module [19] and Comprion test eSIM/eUICC card [20]. Provision of the eUICC is done by the Nokia iSIM secure connect platform [21]. The eUICC is connected indirectly via a modem. Specific AT commands [22] are used to enable the communication between the eUICC and a device. Application Protocol Data Unit (APDU) messages have been defined to interact within the Attestation Applet following the standard ISO7816 [23].

In Figure 7, we present an example of hashes match command message used between the Attestation Applet and the device.

Match Hashes APDU command

CLA	INS	P1	P2	Lc	Data field	Le
0x80	0x50	0x0	0x0	4D	Timestamp + Hashes Values	N/A

The data field contains the timestamp and 2 hashes (77 bytes)

Response APDU

Optional data	Status word	Meaning of status word
N/A	**0x9000**	Successful processing / Match
	0x6305	Invalid Hash / Match

Fig. 7. APDU Command to match hashes

VI. Conclusion & Future Work

Ensuring the integrity of remote devices within distributed ledger technology enabled systems is of paramount importance. The challenge becomes particularly critical when it is employed for service level agreements, where the future settlements, such as, payments, rely on the data sent to distributed ledgers through smart contract executions. Currently, proposed methods require enormous amount of resources and thus are expensive to implement. In this paper, we present DIMSIM, our distributed architecture to ensure device integrity without the need for additional hardware, thereby facilitating inherently secure devices.

We are currently in the process of developing a prototype for DIMSIM and plan to present our analysis in a future work. Importantly, when the future devices are integrated with our solution, their integrity will be guaranteed. Through our solution, the confidence and trust among the stakeholders will be enabled.

REFERENCES

[1] G. Culot, F. Fattori, M. Podrecca, and M. Sartor, "Addressing industry 4.0 cybersecurity challenges," *IEEE Engineering Management Review*, vol. 47, no. 3, pp. 79–86, 2019.

[2] K. A. Demir, G. Döven, and B. Sezen, "Industry 5.0 and human-robot co-working," *Procedia computer science*, vol. 158, pp. 688–695, 2019.

[3] T. Redlich, S. Wulf, M. Moritz, S. Buxbaum-Conradi, P. Krenz, and J. Wulfsberg, "The strategy of openness in industrial production," in *2015 Portland International Conference on Management of Engineering and Technology (PICMET)*. IEEE, 2015, pp. 302–309.

[4] K. Makhijani and T. Faisal, "Accountable and distributed industrial control systems with autonomous contracts : Ocn-dlt: Industry operations and control networks with distributed ledger technology," in *2023 26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2023, pp. 195–202.

[5] T. Faisal, "Accountable and transparent resource sharing and provisioning in future networks," Ph.D. dissertation, King's College London, 2023.

[6] "Unpatched Apache Tomcat Servers Spread Mirai Botnet Malware," https://bit.ly/47nvhmR, accessed: 22-11-2023.

[7] "SGP Embedded UICC Protection Profile," https://bit.ly/GSMAeUICC, accessed: 09-11-2023.

[8] T. Faisal, D. D. F. Maesa, N. Sastry, and S. Mangiante, "How to request network resources just-in-time using smart contracts," in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2021, pp. 1–5.

[9] T. Faisal, M. Dohler, S. Mangiante, and D. R. Lopez, "Beat: Blockchain-enabled accountable and transparent infrastructure sharing in 6g and beyond," *IEEE Access*, vol. 10, pp. 48 660–48 672, 2022.

[10] S. Patil, A. Kashyap, G. Sivathanu, and E. Zadok, "I3fs: An in-kernel integrity checker and intrusion detection file system." in *LISA*, vol. 4, no. 1, 2004, pp. 67–78.

[11] G. H. Kim and E. H. Spafford, "The design and implementation of tripwire: A file system integrity checker," in *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, 1994, pp. 18–29.

[12] D. Chakraborty, L. Hanzlik, and S. Bugiel, "{simTPM}: User-centric {TPM} for mobile devices," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 533–550.

[13] H. Raj, S. Saroiu, A. Wolman, R. Aigner, J. Cox, P. England, C. Fenner, K. Kinshumann, J. Loeser, D. Mattoon *et al.*, "{fTPM}: A {Software-Only} implementation of a {TPM} chip," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 841–856.

[14] N. L. Petroni Jr, T. Fraser, J. Molina, and W. A. Arbaugh, "Copilot-a coprocessor-based kernel runtime integrity monitor." in *USENIX security symposium*. San Diego, USA, 2004, pp. 179–194.

[15] "Altiplano Application Marketplace," https://nokia.ly/47vwiK0, accessed: 22-11-2023.

[16] T. Schläpfer and A. Rüst, "Security on iot devices with secure elements," in *Embedded World Conference, Nuremberg, Germany, 26-28 February 2019*. WEKA, 2019.

[17] "SLM 97CSINFX8000PE," https://bit.ly/Infinon, accessed: 05-10-2023.

[18] "eSIM Whitepaper – The What and How of Remote SIM Provisioning," https://bit.ly/GSMAeSIMWhitePaper, published on: March 2018.

[19] "Quectel LTE EC25 Mini PCIe series," https://www.quectel.com/product/lte-ec25-mini-pcie-series, accessed: 05-10-2023.

[20] "Comprion Test SIMs," https://bit.ly/Comprion, accessed: 05-10-2023.

[21] "iSIM Secure Connect," https://www.nokia.com/networks/security-portfolio/isim-secure-connect/, accessed: 05-10-2023.

[22] "ETSI TS 127 007 v10.3.0 – AT command set for User Equipment (UE) ," https://bit.ly/ETSIaTCommand, published: April-2011.

[23] "ISO/IEC 7816-4-Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange," https://bit.ly/3ZRb1aA, accessed: 05-10-2023.