

This item is likely protected under Title 17 of the U.S. Copyright Law. Unless on a Creative Commons license, for uses protected by Copyright Law, contact the copyright holder or the author.

Access to this work was provided by the University of Maryland, Baltimore County (UMBC) ScholarWorks@UMBC digital repository on the Maryland Shared Open Access (MD-SOAR) platform.

Please provide feedback

Please support the ScholarWorks@UMBC repository by emailing scholarworks-group@umbc.edu and telling us what having access to this work means to you and why it's important to you. Thank you.

Digital Twin Integrity Protection in Distributed Control Systems

Mohammad Ebrahimabadi, Javad Bahrami, Mohamed Younis, Naghmeh Karimi

Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County (UMBC)

Email: {e127, jbahram1, younis, nkarimi}@umbc.edu

Abstract—The notion of Cyber-Physical Systems (CPS) reflects real-time control applications that are realized through distributed coordination among multiple modules. Such coordination is founded on frequent exchange of status and sensor data among the various modules so that actuation decisions are made autonomously. The formation of digital twins has emerged as an effective methodology where data-driven models are employed to enable effective decision making. Hence, the accuracy of these models become very critical for system stability; no wonder data forgery is a major threat for CPS where an attacker strives to inject faulty data to degrade the digital twin of one or multiple modules. Such an attack could be taking the form of impersonating a component, or manipulating/replaying status update packets. This paper proposes an effective scheme for mitigating such a threat by employing hardware-based fingerprinting primitives, namely, Physically Unclonable Functions (PUFs). The proposed PUF-based Integrity protection of digital Twins (PIT) scheme, ensures the authenticity of data sources, and the freshness and integrity of the shared status. PIT is validated using analysis and prototype implementation on an FPGA.

I. INTRODUCTION

A CPS refers to a network of interconnected computational, sensor, and control (actuation) components for managing physical processes. These interconnected components collectively share data to make intelligent decisions and coordinate their actions [1]. Unlike the conventional design of control systems, a CPS advocates decentralized operation in order to scale for large and complex applications. No wonder such a new design methodology has drawn major interest in recent years. Applications of CPS can be found in various domains including, but not limited to, power-grid infrastructures, smart cities, industrial automation, transportation systems, healthcare monitoring, robotic surgical systems, etc.

CPS Security: The security of a CPS is of utmost importance, especially for applications involving public infrastructure [2]. To illustrate, let's consider an impersonation attack in which the adversary masquerades as a legitimate device, or entity within the system. Considering the distributed nature of control and computation in a CPS, such an attack can be highly detrimental as the adversary can deceive the CPS and gain unauthorized access, manipulate the system's operations, or bypass the safety mechanisms [3]. In practice, the impact of such an attack may propagate beyond the compromised node; thus jeopardizing the whole system. Examples of serious incidents include the cyberattack against the Ukraine's power plants in 2015, and the one targeting a Czech hospital during the COVID-19 pandemic causing a shutdown of its IT network and canceling surgeries. However, the scale, complexity of a CPS and the heterogeneity of the involved components make security assurance unconventionally challenging [4].

Digital Twins: The incorporation of digital twins is a new trend for CPS designs, as a means to improve the control process. To elaborate, the advantage of centralized control is attributed to the global knowledge of system-wide parameters. Hence, the ability of distributed control algorithms to reach optimal decisions often increases with growing the scope of inter-component status sharing. A Digital Twin (DT) is a data-driven model that mimics the operation of a component or a subsystem. Consequently, the use of DTs in a CPS allows bridging the gap between the distributed CPS design and centralized approach where a subsystem could infer the status of others using their models (digital twins). Yet, the digital twin itself needs to be continually updated using data from the actual modeled subsystem.

The security challenge is even exacerbated when the CPS exploits DTs [5]. In practice, compromising DTs can lead to malicious control commands, inaccurate simulations, or false insights which in turn can impact decision-making, control actions, and potentially cause physical disruption. Accordingly, securing the DTs helps maintain the integrity and resilience of the overall CPS. In other words, both DT and its physical counterpart (CPS node) need to be robust against various types of disturbances to assure correct functionality [6]. Existing solutions for ensuring DT integrity and detecting anomalies in the context of CPS, can be categorized into data-centric and specification-centric [4]. The former relies on the statistical characteristics of the measurements collected from the various subsystems, while the latter relies on the established standard specifications, and rules to detect deviations from the baseline behavior. Data-centric methods mainly apply Machine-Learning (ML) schemes and are not portable across different CPS domains. On the other hand, specification-centric schemes are challenging as they need the design details to be shared with third parties so they can develop appropriate communication protocols for security monitoring.

Contribution: To overcome the aforementioned shortcomings, this paper proposed, PIT, a novel scheme for verifying the integrity of data and the authenticity of its source in DT-based CPS applications. PIT employs hardware-based fingerprinting primitives, namely PUFs, to associate unique signatures for each data packet that a CPS component transmits. PUFs are hardware security primitives that operate based on the imperfections of the fabrication process of electronic devices. In delay-PUFs (one of the main types of PUFs), such variations cause signals which follow similar paths in the circuit to experience slightly different propagation delays in different chips fabricated from the same design; resulting in a unique

signature per input (so-called challenge) from each PUF [7]. PIT can prevent data forgery attempts by an external adversary who opts to poison the ML-models that DTs apply. PIT does so by leveraging the properties of PUF to counter impersonation and Sybil attacks. For detecting packet replay attacks, PIT combines the PUF-based fingerprints and timestamps. In sum, the contribution of this paper can be summarized as:

- Developing an efficient scheme to ensure data integrity in DT-based CPSs via hardware-based fingerprinting;
- Developing a protocol to tackle message replay attacks in DT-based CPSs by the combination of time- and hardware-based signatures;
- Analyzing the performance of our protocol and its resiliency to contemporary cyberattacks;
- Validating the proposed countermeasures using prototype implementation and testing.

II. RELATED WORK

Modern industrial and infrastructure management applications are characterized by local sensing and distributed control. The potential of DT technology for improving these applications has gained attention in recent years [8], [9]. Despite the importance of security, most of the published studies have focused on the implementation challenges and the design issues in the context of specific applications. For example, S. Khan et al. [10] have developed a framework for DT utilization in power grid management, where DTs are employed to evaluate the remaining useful life of equipment and accordingly scheduling of preventive maintenance. In [11], the authors benefit from digital twins in predicting energy produced from renewable sources such as solar and wind. Meanwhile, DT is used in [12] to optimize the management and sizing of key components within the smart grid, such as heat systems, auxiliary boilers, absorption chillers, etc. Reinforcement Learning is applied to make the DT adaptive to variations in the operating conditions. PIT supports such an adaptive DT operational model where the data of individual CPS nodes are used to update their corresponding DTs. Specifically, PIT protects the data update process against popular cyber-attacks and ensures DT integrity and accuracy.

Although some work has leveraged DTs in the design of distributed control systems in CPS applications, as mentioned above, sustaining the integrity of DTs over time has received little attention. Shen et al. [13] are among the few who published studies that focus on DT vulnerability to data integrity attacks in the context of the micro power grids where sequential hypothesis testing is proposed for detecting such an attack on renewable energy sources. While CPS security is an active area of research [4], existing work does not consider the integrity of DTs, where the physical aspect of the application is exploited to validate the data being used in the decision making, and conventional unicast-type security protocols, e.g., PKI, are employed to secure the communication links. We stress that the DT update process requires a broadcast style of communication to ensure timeliness and DT consistency across the CPS system. Although PUFs have been extensively used in recent years for security provision [14], thanks to their low overhead and unclonability, published PUF-based

protocols are mainly geared for unicast communications. In this paper, we deploy such a hardware primitive to secure broadcast traffic. Hence, PIT indeed fills a technical gap.

III. SYSTEM MODEL AND PRELIMINARIES

A. System and Threat Models

PIT considers a CPS that comprises N nodes, each equipped with sensing, processing, and actuation capabilities. The operation model of such a CPS is to monitor globally (system wide) and act locally (at the subsystem level). Hence, each node includes the DT of all other $N-1$ nodes; such DT is used by the local (node-level) control algorithm. In other words, a CPS node CN_i , considers $DT_j, \forall j \neq i$ ($1 \leq i, j \leq N$) in order to make appropriate decisions and determine adjustments of local control variables without conflicting with other subsystems, i.e., other CPS nodes. The CPS nodes are interconnected through wireless links, where each node CN_i broadcasts its data to all other nodes to update and synchronize the corresponding DT_i used by these nodes.

PIT strives to counter an external adversary who opts to diminish the accuracy of the employed DTs. To achieve such a goal, the adversary tries to inject faulty data to poison the underlying ML model of the individual DTs. The attack fundamentally tries to get a CPS node, CN_i , to base its action on its inaccurate view of the other nodes, represented by their DTs on CN_i . By using inaccurate or even faulty DTs, CN_i could cause not only local failures but also could bring down the whole system. As means to reach such a malicious goal, the adversary attempts: (1) impersonating one or multiple CPS nodes (the latter is called Sybil attack) so that wrong data can be injected without suspecting the source, (2) replaying packets to hinder model update so that the DT does not reflect recent changes in the node behavior, resulting in inappropriate decisions, and (3) eavesdropping on the communication links between nodes to intercept transmissions and launch a Man-In-The-Middle attack to manipulate the shared data. In summary, PIT is to mitigate *Impersonation*, *Sybil*, *Message replay*, and *Man-In-The-Middle* attacks.

B. Preliminaries

Digital Twins: A DT refers to the virtual duplicate of a physical object, process, or system [15]. It mimics the behavior of the original replica in real-time. In a CPS, the DT of each node is built based on the data collected from sensors, actuators, etc. ML is usually used as the underlying modeling methodology. Synchronization between the physical node and its virtual representation (i.e., twin) is necessary, especially in a CPS, in order to cope with the dynamic nature of data in these systems. In practice, a DT should be constantly updated in real-time based on the status messages (data) from the replicated physical object. Therefore, reinforcement learning techniques are often employed. Accordingly, a DT is able to accurately predict the behavior of the physical counterpart.

Physically Unclonable Functions: A PUF is a hardware-based security primitive that is both unclonable and unpredictable. The utility of a PUF is founded on its ability to generate unique signatures that can be deployed for device authentications or generating keys and random numbers for cryptographic

algorithms. A PUF design takes advantage of the random variations in the manufacturing process of electronic devices. The arbiter-PUF is deemed to be the most popular PUF design given its large number of input-output combinations (challenge-response pairs), which has made it an attractive choice for supporting device authentication [16]. An arbiter-

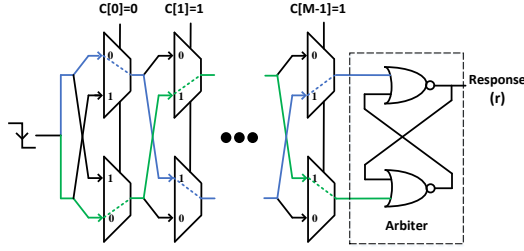


Fig. 1. Illustrating the design of an arbiter-PUF.

PUF consists of pairs of identical paths (e.g., blue and green paths shown in Fig. 1); the manufacturing process variations introduce a race that corresponds to the difference in signal propagation delay on these two paths, and affects the value latched by the arbiter. The arbiter can be realized as a simple SR-latch implemented by two cross-coupled NOR gates. Note that each PUF instance generates a 1-bit response (shown as r in Fig. 1). A multi-bit response (denoted with R hereafter) is extracted either by instantiating multiple PUFs in the hardware or by embedding only one PUF sample yet querying it multiple times with different challenges. In this paper, we consider the latter case to lower the area overhead.

IV. PIT METHODOLOGY

PIT employs PUFs as a means for fingerprinting CPS nodes. As mentioned earlier, the response of a PUF to a challenge bit-string is device-specific and varies from one PUF to another. Thus, a Challenge-Response Pair (CRP) of a PUF can be used as a signature for the device in which the PUF is embedded. Specifically, in PIT each CPS node CN_i ($1 \leq i \leq N$) will incorporate a PUF with a large number of CRPs, such as the arbiter-PUF. As explained in the balance of this section, the PUF's CRPs will be used as fingerprints not only for authenticating CN_i when it sends data to other nodes but also for ensuring the integrity of the transmitted data.

PUF-based Authentication and Data Protection: Given the operation model of a distributed control system, every CPS node CN_p will share its data/status with others. The device that has the PUF, CN_p in this case, is referred to as a prover while the other end of the communication link is referred to as a verifier, i.e., CN_v ($v \neq p$). Hence, as a prover, CN_p will use its PUF to generate secrets to protect its data transmission and to prove its identity. One of the main advantages of a PUF is that the response to a challenge bit-string is generated by the prover in real-time rather than being stored. However, the verifier needs to know the secret that the prover employed in order to conduct authentication and retrieve the data that CN_p included in a packet. Consequently, at the time of system deployment CN_v is provided with a subset of the CRPs of CN_p . An alternative option is to provide a digital twin of the PUF of CN_p to other CSP nodes. Such a DT is in essence a machine learning based model of the PUF. In that case, CN_v does not need to store

the CRPs of CN_p and instead generates the response using the provided PUF's digital twin.

As mentioned earlier, each node CN_p broadcasts its data/status to update and synchronize the corresponding DT_p used by other CPS nodes. In PIT, CN_p is authenticated each time it sends a data packet. To prevent data leakage we obfuscate the data being sent yet not using symmetric or asymmetric cryptographic schemes, considering their drawbacks in the context of resource-constrained environments. Basically, we use the PUF's CRP to generate a key that is combined with the packet payload using simple Boolean operations. To mitigate the vulnerability of such a simple data obfuscation scheme to cryptanalysis, the utilized CRPs are varied as frequently as per packet. Meanwhile, to detect packet replaying attacks, we incorporate a timestamp in each packet's payload, as explained later in this section.

To tackle the impersonation attacks, PIT benefits from the unique signatures provided via the embedded PUF in each node. However, to reduce the overhead imposed on the PIT for storing the CRPs of a node CN_i at all other nodes, only a subset of those pairs will be used. The size of the subset is subject to trade-off. If a small subset is considered, an adversary who intercepts the transmissions made by CN_i can replay them, thus impersonating CN_i . As mentioned earlier, such a threat can be tackled via the inclusion of a timestamp along with the CRPs in the payload. To prevent data leakage as well as impersonation attacks in case a CRP is leaked we employ a fresh key where a random number is generated and used (as discussed below) to encrypt the payload.

Data Packet Formation: In PIT each node CN_i ($1 \leq i \leq N$) broadcasts its data to all other nodes; the packet format is defined in Eq. (1), where the symbol \parallel represents concatenation and MAC_j denotes the Message Authentication Codes (MAC) for the receiver node of CN_j . These MACs are appended to the payload data and the packet is sent by CN_i to all other nodes to update their individual copies of DT_i , i.e., update the digital twins of CN_i at each packet receiver CN_j .

$$Packet = \parallel_{j=1, j \neq i}^N MAC_j \parallel Payload \quad (1)$$

The individual MAC values generated by the sender CN_i are determined using Eq. (2) where $RND_{i,t}$ is a random number generated by CN_i at the time t when the packet is formed. $RND_{i,t}$ serves as an encryption key for the data payload, although no elaborate cryptographic algorithm is used and simple XOR operations are used instead. The function f in essence opts to obfuscate the encryption key so that it can be extracted only by the intended receiver. To do so, $f(PUF_i(C), ID_j)$ combines the PUF response to a challenge C with the ID of the verifier (packet receiver) CN_j . Factoring in the ID of CN_j opts to make the MACs distinct among the different receivers and ensure that only specific CPS nodes retrieve the data. The value of C is picked by the data source, i.e., CN_i , and could vary per packet. The function f should be lightweight, yet should prevent an eavesdropper from inferring $R = PUF_i(C)$ from the MACs in the packet so that CRPs cannot be aggregated over time to form a model of PUF_i .

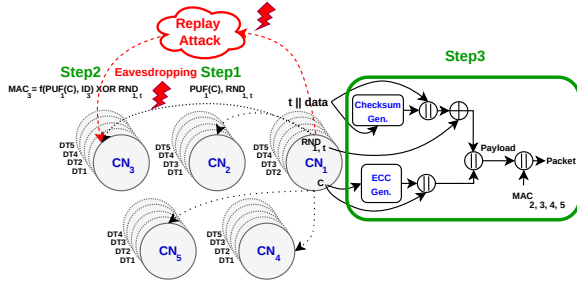


Fig. 2. Summarizing the Data Packet Formation in PIT

using machine learning techniques. In other words, the ID should not simply be used as a mask. An example of the function f could be $PUF_i(C) \oplus PUF_i(ID_j)$.

$$MAC_j = f(PUF_i(C), ID_j) \oplus (RND_{i,t}) \quad (2)$$

In PIT, the payload is formed by Eq. (3). The challenge C is included to enable the receiver to know how to decode the packet payload. Note that only an authorized CPS node knows the challenge's response, i.e., $R = PUF_i(C)$. Again, t denotes the time when the packet is formed. The granularity of t can be different based on the application and the frequency of sending data between CPS nodes so that message replaying can be detected while limiting the packet length overhead. For example, if data packets are sent at the rate of once per minute, there is no need to have a time resolution in seconds. In Eq. (3), $Data_t$ refers to the data that CN_i wants to broadcast. Moreover, the checksum, chk , of the concatenation of the time stamp (t) and $Data_t$ is also appended to these two pieces of information in order to ensure integrity.

$$Payload = C || ECC || (RND_{i,t} \oplus (t || Data_t || chk)) \quad (3)$$

A PUF response to a specific challenge may be affected by variations in the operating temperature or voltage noise in the underlying device. Moreover, a digital twin of a PUF, if used, may not be 100% accurate. To correct the PUF response in these cases, Error Correction Codes (ECCs) are often used. Since, the error rate of PUF responses due to noise is less than 2% [17] and due to PUF model inaccuracy, if indeed used, is around 1% [7], only few ECC bits are needed. For example, in a 64-bit response, which is more than enough for most CPS applications, at most 3 bits may change due to the temperature and voltage variations, or the imperfection of the PUF's digital twins based on the aforementioned statistics. Therefore, a low-cost ECC to correct 2-3 bits would be appropriate for PIT. Accordingly as shown in Eq. (3), PIT also sends the ECC as part of the payload. The entire packet formation process is illustrated in Fig. 2. In the figure, CN_1 is the sender node. Each node has the DT of other nodes, represented by dotted line circles. Potential attack points are highlighted in red.

Receiver-Side Verification and Data Retrieval: After CN_i forms the packet using Eq. (1), the packet is transmitted. On the receiver side, the packet is decoded to retrieve the data as well as authenticate the sender node CN_i . First, a receiver CN_j retrieves its related MAC_j along with the payload from the received packet using Eq. (1). By referencing the tabulated PUF response (or using the digital twin of PUF_i),

node CN_j can extract $RND_{i,t}$ from MAC_j using Eq. (2). Specifically, CN_j should extract C from the packet payload, reference the tabulated CRPs or the DT of the PUF of CN_i to get $PUF_i(C)$; for that the ECC should be applied to make any necessary correction. Then, CN_j will factor in ID_j to calculate $f(PUF_i(C), ID_j)$. ID_j may be involved in finding out $PUF_i(ID_j)$ as noted earlier and in such a case the ECC for $PUF_i(ID_j)$ should be appended to the MAC_j as well.

The retrieved $RND_{i,t}$ is then used to extract $t || Data_t || chk$ from the packet payload, as specified in Eq. (3). Upon obtaining timestamp (t) and $Data_t$, their values are validated using the checksum chk to ensure integrity, i.e., the $Data_t$ has not been changed due to man-in-the-middle-attack, transmission error, or uncorrected PUF response (due to the occurrence of more bit flips in the PUF output than anticipated). It is important to note that the ECC value semantically does not correct the error in the PUF output but rather enables the receiver to know the value of $PUF_i(C)$ that the sender used. To elaborate, assume that at the time of system deployment CN_j is informed that $PUF_i(C) = R_{perfect}$, yet during operation the output of CN_i 's PUF happens to be noisy and R_{noisy} is used in forming the packet. In such a case, CN_j will use the included ECC in the packet to transform $R_{perfect}$ to R_{noisy} since the latter is what CN_j actually used.

V. PERFORMANCES ANALYSIS

We compare the PIT's overhead in terms of energy and latency with the cases where conventional symmetric or asymmetric cryptographic algorithms are used to protect the data transfer between CPS nodes. To perform such analysis, the CPS nodes are assumed to employ *Jennic JN5139* communication modules [18]. Such a module utilizes an IEEE 802.15.4/ZigBee transceiver operating in a voltage range of 2.3V to 3.6V, with an output power of 2.5 dBm. To factor in the typical proximity among CPS nodes, we consider an output power of 1 dBm (equivalent to 1.2 mW). We estimate the communication-related power consumption assuming a network of 5 nodes ($N = 5$) and a packet header of 4 Bytes. **Energy:** For a 2.9V power supply and drawn current of 15 mA and 17.5 mA during transmission (T_x) and reception (R_x), respectively, the energy overhead for a JN5139 module is:

$$\begin{aligned} T_x \text{ Power} &= (2.9 \times 15) + 1.2 = 44.7 \text{ mW} \\ R_x \text{ Power} &= (2.9 \times 17.5) = 50.75 \text{ mW} \end{aligned} \quad (4)$$

Considering an IEEE 802.15.4/ZigBee transceiver, the maximum raw data throughput is 250 Kbits/s. Thus:

$$\begin{aligned} \text{Energy per } T_x \text{ Bit} &= \frac{44.7 \text{ mW}}{250,000 \text{ bit/s}} \approx 179 \text{ nJ/bit} \\ \text{Energy per } R_x \text{ Bit} &= \frac{50.75 \text{ mW}}{250,000 \text{ bit/s}} \approx 203 \text{ nJ/bit} \end{aligned} \quad (5)$$

The PIT prototype implementation discussed in Sec. VI, uses an arbiter-PUF with 64-bit challenge and response. To correct the noise impacts on PUF responses, we deploy the row-column parity scheme explained in [19]; thus a 16-bit ECC is needed. The MAC and $RND_{i,t}$ are 64 bits each. We assume the combined size of t , $Data_t$, and $Checksum$ in Eq. (3) is also 64 bits. Therefore, the size of a packet consisting of 4

MACs (as we assumed that the CPS includes 5 nodes) along with the payload would be 54 bytes assuming that we have a 4-byte header for each packet. Hence the energy for each transmitted and received packet is:

$$\begin{aligned} \text{Energy} / T_x &= \frac{179 \times 54 \times 8}{1000} \approx 0.077 \text{ mJ} \\ \text{Energy} / R_x &= \frac{203 \times 54 \times 8 \times (5 - 1)}{1000} \approx 0.350 \text{ mJ} \end{aligned} \quad (6)$$

Next we compare the overhead of PIT with authentication schemes that use symmetric encryption. It is known that asymmetric algorithms are resource intensive. Hence, we only compare PIT with AES (128-bit key). When using AES, every two nodes set a unique key for encryption/decryption, i.e., a total of 10 keys for the considered 5-node CPS. The energy consumption of an AES core varies depending on its implementation, typically ranging from 5 mJ to 34 mJ [20]. For the considered DT-based CPS, each node broadcasts one message and receives $N - 1$ data update messages. AES does not support broadcast and requires each node to transmit $(N - 1)$ unicast messages. Therefore, an AES core alone imposes a minimum energy cost of $2 \times (N - 1) \times 5 \text{ mJ} \approx 40 \text{ mJ}$ for $N = 5$. This is in addition to the energy consumed in communication. Meanwhile (based on the report from Xilinx Vivado) our implemented PUF power is less than 2 mW, and considering the 17 μs PUF query time, the implemented 64-bit arbiter PUF consumes around 34 nJ, which is quite negligible. Thus, PIT is significantly more energy efficient than AES-based schemes. It is worth noting that PIT also outperforms stream-cipher based approaches, where a lightweight LFSR-based stream cipher, such as Trivium, consumes approximately 81 mJ on a single-board micro-controller IoT platform [21].

Latency: We compare PIT's latency with that of conventional encryption schemes. PIT's packet includes $(N - 1)$ MACs, a 4-byte header, and a payload. For a 64-bit arbiter PUF, each MAC would be 64-bit. For C , ECC , and RND being 64, 16, and 64 bits, respectively, the payload size is $2 \times 64 + 16$ bits (Eq. 3). Thus each node broadcasts $[64 \times (N + 1) + 48]$ bits. When using 128-bit AES, the packet size almost doubles to become $(N - 1) \times 128$. The time for AES to encrypt a 128-bit block on a Raspberry Pi platform is between 28.6 and 108.5 ms [20]. PIT involves PUF query and simple XOR operations (XOR latency is in the order of μs). Please note that to make a 64-bit response we need to query the PUF 64 times; however in the worst case scenario, the latency of generating 64-bit response is in the order of μs . Finally, based on the recent study in [22], a lightweight LFSR-based stream cipher takes $\approx 76 \text{ ms}$ to encrypt 256 bytes of data on an Arduino platform with 16 MHz ATmega328P microcontroller and 32 KB RAM. Recall that for PIT, the length of one packet is $64 \times (N - 1)$ bits as we need to encrypt data for each receiver, separately. Therefore $64 \times (N - 1) \times 0.037 \text{ ms} \approx 9.5 \text{ ms}$ (for $N = 5$) is required to generate such a packet. Both PIT and stream cipher take the same time to send a packet. Hence, PIT significantly outperforms conventional encryption schemes.

VI. EXPERIMENTAL SETUP & RESULT

FPGA-based Implementation: In this section, we describe the implementation of PIT on real silicon, specifically using a

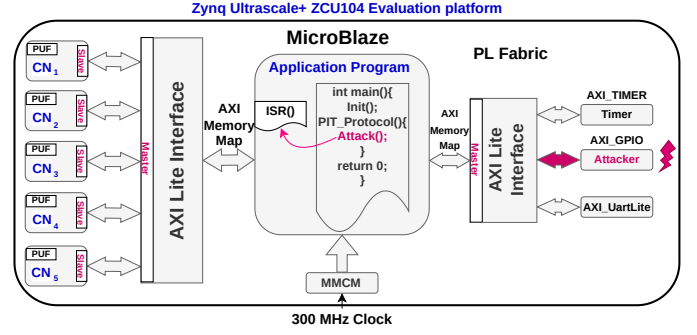


Fig. 3. FPGA implementation of PIT (Zynq Ultrascale+ ZCU104 Platform)

Zynq Ultrascale+ ZCU104 Evaluation platform equipped with a Zynq Ultrascale+ FPGA. Instead of using separate nodes, we pursued a hardware/software co-design methodology to implement the entire network of five nodes on a single FPGA, as illustrated in Fig. 3. Each node was represented as an Intellectual Property (IP) written in VHDL, incorporating its respective PUF. To facilitate our implementation, we utilized the *AXI_Lite* protocol, as it allowed every node to update its device token every 0.89 ms, in line with the system's frequency of 80 MHz. Due to its low overhead and complexity, this choice was preferred over *AXI_Stream* and *AXI_Full*.

The PUFs were carefully placed on the FPGA using manual placement and macro creation in Xilinx Vivado 2023.1 to ensure similar delays between various multiplexers (implemented using LUTs). To execute the application, we employed the MicroBlaze Xilinx soft processor in Vitis 2023.1 environment, equipped with 128 KB of internal memory. For time measurements, we integrated an AXI-Timer into the system. Additionally, we employed AXI-UartLite for serial communication with the HOST and for debugging purposes. To model attack scenarios, we utilized AXI-GPIOs with push buttons, utilizing interrupts managed by the AXI-INTC IP to handle interrupts sent to MicroBlaze. The entire process starts by generating random numbers (using an LFSR implemented within the application using C programming language), Challenge (C), and the current system time (clock count).

MACs were then generated using Eq. (2) for all nodes, and the latency was measured. Our results demonstrated that the entire PIT protocol took 0.89 ms ($12.5 \text{ ns} \times 71,627$ clock cycles). Such latency is quite low and suits many DT-related CPS applications. For example, in a smart power grid PMUs transmit voltage and current phasors measurements at a rate of 60 samples per seconds, i.e., every 15 ms, and hence PIT would not constitute much overhead. Also, every node in our implementation, considering the internal PUF and its associated controller, takes 255 LUTs and 283 FFs on Zynq Ultrascale+ ZCU104 Evaluation platform. The entire system resource utilization is also shown in Table I.

TABLE I
RESOURCE UTILIZATION OF PIT ON ZYNQ ULTRASCALE+ FPGA

| Metric | LUT | SRL | Flip-Flop | Block RAMs | MMCM |
|--------|-------|-----|-----------|------------|------|
| No. | 5,107 | 322 | 6,192 | 35 | 8 |

Attacks Implementation and PIT outcome: We refer to the implemented CPS nodes as CN_1, \dots, CN_5 . We assumed that CN_3 might act normally or maliciously. For the latter, we

implemented three attack scenarios and incorporated three dip switches to selectively enable them. The following describes these attacks and the observed behavior (when applying PIT):

- **Impersonation & Sybil Attacks:** The dip switches were set to designate CN_3 as a rogue node that tried to impersonate CN_4 in our setup. CN_3 knows how to generate a valid packet in PIT, and also has all information regarding packet header, ID, etc. of CN_4 . Therefore, CN_3 forms the packet based on the information of CN_4 , yet it uses its own PUF output (since no node knows the response of the PUF of CN_4). CN_3 then forwards the generated packet to the other nodes (MicroBlaze in our implementation). Upon receiving the packet, each of nodes CN_1 , CN_2 , and CN_5 , decoded the packet to extract the challenge bits string, C (Eq. (3)), and mapped it to the corresponding PUF response of CN_4 , denoted R_4 (recall Eq. (2)). Following the steps in Section IV, each recipient applies ECC, and then uses the curated R_4 to extract RND from the MAC . Such RND does not match what CN_3 used since the employed CRP during the packet formation is not for CN_4 . When the inferred value of RND was used for extracting the data, the checksum was wrong and the packet was discarded. The experiment was then repeated where CN_3 tried to impersonate both CN_4 and CN_2 , to launch a Sybil attack. In such a case, nodes CN_1 and CN_5 , successfully detected the message payload inconsistency and discarded the two malicious packets.
- **Man-in-the-middle-Attacks:** To validate PIT's ability to detect data manipulation, we set the dip switches to trigger the second attack scenario, where the generated packet by CN_3 is corrupted via replacing some of the bits in its payload with randomly selected values of '1's and '0's. PIT could successfully detect such an attack during the decryption step, as explained in Section IV, since the checksum generated by a receiver, say CN_4 , does not match the checksum that CN_3 embedded in the packet.
- **Packet Replay Attacks:** We configured the dip switches to mimic packet replay. As shown in Eq. (3), each packet payload includes the time t referring to the time when the packet is formed as well as a fresh random number generated at time t . In our implementation, both of these values are provided by AXI-Timer module. In this experiment, a packet is generated and broadcasted by CN_3 . Upon reception, each node decrypts such a packet and stores the value of t in a register (say REG) to be able to track the timestamp of the most recently generated packet by CN_3 . PIT could easily detect a packet replay during decoding where the packet payload is decrypted and the timestamp is extracted. If such timestamp does not exceed that of the previously received packet (stored in REG), the packet is deemed to be a replay and is discarded.

In summary, the prototype based testing has confirmed PIT's effectiveness in detecting such cyberattacks.

VII. CONCLUSION

Incorporation of digital twins is being viewed as a viable technique for the realization of distributed control systems, where decisions could be made locally without violating the

stability of the overall system. Given such data-driven design, the accuracy of the DT model is of utmost importance and data forgery can be a major threat to the integrity of the system. This paper has presented a novel protection mechanism, namely PIT, for detecting malicious attempts to degrade the DT accuracy by injecting malicious data to retrain the underlying ML models. PIT benefits from both hardware- and time-fingerprinting to counter attempts to provide faulty data by launching impersonation, man-in-the-middle and message replay attacks. Prototype implementation on an FPGA has validated PIT's ability in achieving all its design goals.

REFERENCES

- [1] L. Ribeiro and M. Björkman, "Transitioning from standard automation solutions to cyber-physical production systems: An assessment of critical conceptual and technical challenges," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3816–3827, 2018.
- [2] A. Humayed et al., "Cyber-physical systems security—a survey," *IEEE IoT-J*, vol. 4, no. 6, pp. 1802–1831, 2017.
- [3] M. Ebrahimabadi et al., "Hardware assisted smart grid authentication," in *ICC*, 2021, pp. 1–6.
- [4] H. Lin et al., "Challenges and opportunities in the detection of safety-critical cyberphysical attacks," *Computer*, vol. 53, no. 3, pp. 26–37, 2020.
- [5] M. Eckhart et al., "Digital twins for cyber-physical systems security: State of the art and outlook," *Security and Quality in Cyber-Physical Systems Engineering: With Forewords by Robert M. Lee and Tom Gilb*, pp. 383–412, 2019.
- [6] C. Gao, H. Park, and A. Easwaran, "An anomaly detection framework for digital twin driven cyber-physical systems," in *Int'l Conf. on Cyber-Physical Systems*, 2021, pp. 44–54.
- [7] U. Rührmair et al., "PUF modeling attacks: An introduction and overview," in *Design, Automation & Test in Europe*, 2014, pp. 1–6.
- [8] M. Jafari et al., "A review on digital twin technology in smart grid, transportation system and smart city: Challenges and future," *IEEE Access*, vol. 11, pp. 17 471–17 484, 2023.
- [9] Q. Nie et al., "A multi-agent and cloud-edge orchestration framework of digital twin for distributed production control," *Robotics and Computer-Integrated Manufacturing*, vol. 82, p. 102543, 2023.
- [10] S. Khan et al., "Digital twin for advanced automation of future smart grid," in *ICAISC*, 2023, pp. 1–6.
- [11] L. You and M. Zhu, "Digital twin simulation for deep learning framework for predicting solar energy market load in trade-by-trade data," *Solar Energy*, vol. 250, pp. 388–397, 2023.
- [12] Q. He et al., "Management and real-time monitoring of interconnected energy hubs using digital twin: Machine learning based approach," *Solar Energy*, vol. 250, pp. 173–181, 2023.
- [13] Z. Shen et al., "Digital twin application for attach detection and mitigation of pv-based smart systems using fast and accurate hybrid machine learning algorithm," *Solar Energy*, vol. 250, pp. 377–387, 2023.
- [14] K. Lounis and M. Zulkernine, "Lessons learned: Analysis of puf-based authentication protocols for iot," *Digital Threats*, vol. 4, no. 2, Feb 2022.
- [15] S. Mihai. et al., "Digital twins: A survey on enabling technologies, challenges, trends and future prospects," *Communications Surveys Tutorials*, vol. 24, no. 4, pp. 2255–2291, 2022.
- [16] A. Shamsoshoara et al., "A survey on physical unclonable function (puf)-based security solutions for internet of things," *Computer Networks*, vol. 183, p. 107593, 2020.
- [17] U. Chatterjee et al., "Building puf based authentication and key exchange protocol for iot without explicit crps in verifier database," *TDSC*, vol. 16, no. 3, pp. 424–437, 2018.
- [18] "Product brief-jn5148 module (jennet.zigbee pro and ieee802.15.4 module)," <https://www.glynstore.com>, 2010.
- [19] M. Ebrahimabadi et al., "Sweet: Security protocol for wearables embedded devices' data transmission," in *HealthCom*, 2022, pp. 135–141.
- [20] B. Tsao et al., "Analysis of the duration and energy consumption of aes algorithms on a contiki-based iot device," in *Int'l Conf. on Mobile and Ubiquitous Sys.: Comp., Net. and Services*, 2019, pp. 483–491.
- [21] L. Ertaul and A. Woodall, "IoT security: Performance evaluation of Grain, MICKEY, and Trivium - lightweight stream ciphers," in *IEEE Conf. on Security and Management*, 2017, pp. 32–38.
- [22] B. B. S. Deb, "Performance analysis of current lightweight stream ciphers for constrained environments," *Sādhanā, the Indian Academy of Sciences*, vol. 45, no. 256, 2020.