Proc. of International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA 2025)
7-9 August 2025, Antalya-Türkiye

Approximating Quantified Information Leakage in Cyber-Physical Systems Through Their Digital Twins

Sepehr Amir-Mohammadian
Dept. of Computer Science
University of the Pacific
Stockton, CA, USA
samirmohammadian@pacific.edu

Khayyam Salehi
Dept. of Computer Science
Shahrekord University
Shahrekord, Iran
kh.salehi@sku.ac.ir

Afsoon Yousefi Zowj
Dept. of Computer Science
University of the Pacific
Stockton, CA, USA
ayousefizowj@pacific.edu

***Abstract*—Quantitative information flow analysis provides a systematic approach to evaluating the security of a system by measuring the amount of secret information exposed through public outputs. In this paper, we develop a formal framework for quantifying information leakage in cyber-physical systems by modeling their digital twins. We define a process-algebraic calculus for digital twins and specify their semantics using Markovian models, enabling rigorous information flow analysis. Cyber-physical systems operate in an uncountably infinite state space due to their continuous physical behaviors. In contrast, digital twins—through digitization—evolve in a finite space, enabling the feasibility of quantitative information flow analysis using Markovian models. This formulation captures the inherent nondeterminism and probabilistic transitions in cyber-physical systems arising from evolution uncertainties, sensor measurement errors, and concurrent control logic. Leveraging back-bisimulation, we systematically compare an attacker's knowledge of system secrets before and after execution to quantify leakage, relying on Renyi's min-entropy. This approach provides a structured methodology for assessing confidentiality risks in cyber-physical systems and demonstrates the role of digital twins in advancing security analysis.**

***Keywords*—Large Language Models, Network Protocols, Request for Comments, Testing**

I. Introduction

Cyber-Physical Systems (CPSs) are integrations of computational and physical processes, where embedded software interacts closely with physical components through sensors and actuators. These systems are designed for real-time operations, offering precise control and feedback mechanisms. CPS features include tight integration of cyber and physical elements, real-time operation, adaptability, resilience, and the ability to operate autonomously or collaboratively across networks [1].

CPSs have become pervasive in today's world, powering smart grids, autonomous vehicles, and industrial IoT applications. For instance, autonomous cars rely on CPSs to integrate real-time data from sensors to navigate safely [2]. Similarly, smart manufacturing systems employ CPSs to optimize production lines dynamically [3]. Their prevalence continues to grow with advancements in edge computing and artificial intelligence, making them essential to modern technological ecosystems [4]. For instance, edge computing has enabled real-time data processing at the source, which is crucial for applications like autonomous drones that rely on immediate responses for navigation and obstacle avoidance [5]. Similarly, AI-powered predictive maintenance systems in industrial CPSs reduce downtime by identifying potential failures before they occur [6].

CPSs are increasingly targeted by attackers exploiting the cyber domain to gain insights into sensitive physical components. In this context, attackers often employ inference attacks, where they analyze the behavior of cyber components (e.g., network traffic, sensor readings, or control signals) to infer secrets about the physical system's operations or parameters [7]. These attacks differ from side-channel attacks, which exploit unintended physical manifestations (e.g., heat, sound, or electromagnetic emissions) to deduce information [8].

Models of CPSs inherently involve uncountably infinite state spaces, primarily due to the continuous nature of the physical components they represent. Examples include models based on automata theory, e.g., [9], hybrid-dynamic models, e.g., [10], and process algebraic models, e.g., [11], of CPSs. In particular, Lanotte et al. [12] utilize process algebra, a variant of TPL [13], to model CPSs using discrete-time labeled transition semantics, known as the Calculus of CPSs (CCPS). In this framework, each state in the system potentially transitions to an uncountably infinite number of other states, as the parameter values are drawn from continuous real-number ranges. Although CCPS offers a robust and suitable core linguistic model for CPSs, it is limited in its application for quantifying information flow within CPSs. This uncountable infinity of transitions arises from two sources: *evolution uncertainty* and *sensor measurement error*. Evolution uncertainty defines a range within which each variable can fluctuate in the target state, resulting in a continuous space of possible values. Similarly, sensor measurement error defines a range of possible measured values for a parameter of the physical plant due to inaccuracies in measurement, creating another continuous span of real numbers. These combined sources of uncertainty contribute to the complexity of analyzing CPSs.

The uncountable nature of the state transitions in CPS models complicates the precise and manageable quantification of information flow. The inherent complexity of continuously varying parameters within these models makes it difficult to directly apply them to information leakage quantification tasks. Given this restriction, in this paper, we are encouraged to shift our focus to digital twins (DTs) of CPSs and explore approaches to measure information leakage within these systems. DTs provide a manageable and discrete abstraction of CPSs, making them a more practical candidate for addressing the challenges posed by uncountable state transitions in information flow quantification.

DTs are virtual replicas of physical systems that may mirror their real-world counterparts through continuous data exchange. This synchronization enables real-time monitoring, analysis, and optimization of systems across various domains, including manufacturing, healthcare, transportation, and energy [14]. For instance, in manufacturing, DTs facilitate predictive maintenance and process optimization by simulating production lines and identifying potential issues before they occur [15].

In the context of CPS security, DTs serve as a critical tool for enhancing resilience against cyber threats. By creating a virtual environment that mirrors the physical system, DTs allow for the safe testing and validation of security measures without risking actual operations. They can simulate attack scenarios, assess vulnerabilities, and develop effective countermeasures [16]. Moreover, DTs can monitor and analyze sensor data to detect security threats or unsafe conditions in real time, triggering appropriate responses to mitigate risks [17]. DTs provide a structured framework that could be used to analyze how sensitive data propagates through system components. By simulating information flow, DTs can help identify potential leakage points and evaluate the effectiveness of mitigation strategies. For example, in a recent study, DTs were employed to simulate data flows in an industrial control system, successfully pinpointing a vulnerable data exchange channel that could lead to information leakage [18].

It is worth noting that for the purposes of this work, we treat DTs as simulations of CPSs, focusing on their role as analytical tools rather than their real-time synchronization capabilities. This assumption aligns with the scope of the study, allowing us to leverage DTs as simplified models for analyzing and quantifying information leakage.

We present a formal calculus for DTs that produces a finite set of successor states at each stage of the system's evolution. This finite design makes our information leakage quantification tractable by keeping the state space manageable when modeled as Markovian processes. Leveraging the resulting Markovian models, we introduce a systematic method to quantify the amount of leaked information through probabilistic reasoning over state transitions. Through this approach, we effectively approximate the quantification of information leakage in CPSs by analyzing their corresponding DTs. Here, approximation refers to the digitization of continuous physical behaviors, such as real-valued sensor readings or environmental parameters, into finite-precision representations (e.g., floating-point values), making CPS semantics amenable to discrete-state analysis.

Quantitative information flow (QIF) [19] is crucial in cybersecurity as it provides a systematic way to measure the amount of secret information leaked through system outputs, enabling the evaluation of partial confidentiality breaches. This approach is essential for balancing practicality and security, as some minor leakages are inevitable in real-world systems. QIF is widely applied in areas like timing attacks [20], [21], differential privacy [22], [23], and cryptographic analysis [24], making it a fundamental tool for assessing and mitigating information exposure in diverse security contexts.

Assume a DT with a secret input h and a public output l. An attacker, with full knowledge of the DT's cyber component, executes the DT and observes l. Information leakage is quantified using the notion of *knowledge uncertainty*: **leaked information = initial knowledge uncertainty - remaining knowledge uncertainty** [25]. Before execution, attacker's knowledge uncertainty about h is determined by her prior knowledge. Observing the output during execution reduces this knowledge uncertainty, revealing information about h.

This paper introduces a formal approach to quantify information leakage in terminating DTs, by incorporating intermediate state leakages and the impact of scheduling policies. The secret input h remains constant during execution, while l and neutral variables (neither confidential nor public) are initialized to single values. The attacker is probabilistic, can select a scheduler, observe execution traces of l, and execute the DT arbitrarily many times under a one-try guessing model [25]. This setup aligns with common confidentiality analyses that assume data integrity is out of scope, e.g., [26]–[28].

To model DTs for QIF analysis, Markov Decision Processes (MDPs) are employed, offering a robust framework for modeling probabilistic and nondeterministic behaviors [29]. A probabilistic scheduler resolves the nondeterminism in the MDP, inducing a Markov Chain (MC). MC states capture values of h, l, and neutral variables. However, for leakage computation, the MC must reflect the attacker's perspective. The attacker, while observing execution traces, cannot determine the exact value of h but can infer a set of possible values based on observed traces and executed steps. Moreover, the attacker cannot distinguish between MC executions that yield the same trace. To address this, we use a back-bisimulation equivalence relation [28], which induces a quotient MC that models the attacker's view of the DT.

In the back-bisimulation quotient, each state is associated with a distribution, representing the possible values of h in that state and determining the attacker's knowledge uncertainty about h. Execution traces of the quotient represent the reduction in knowledge uncertainty from the initial state to the final state. The initial state's distribution defines the attacker's initial knowledge uncertainty, while final states determine the remaining knowledge uncertainty. Following

Smith [25], knowledge uncertainty in the one-try guessing model is quantified using *Renyi's min-entropy*, which measures the difficulty of guessing h. Information leakage is thus computed as the difference between the Renyi's min-entropy of h in the quotient's initial state and the expected Renyi's min-entropy of h in its final states.

Contributions. The key contributions of this work are as follows:

- **Formal calculus for DTs:** We introduce a process-algebraic calculus with labeled transition semantics tailored for DTs, enabling precise modeling of CPS behavior in a digitized, finite state space.
- **Markovian semantics for information flow Analysis:** By modeling DTs as MDPs and resolving nondeterminism via probabilistic schedulers, we formalize their execution as MCs, suitable for quantitative reasoning.
- **Back-bisimulation for attacker's view:** We apply back-bisimulation to construct the attacker's view of the DT's MC, enabling accurate leakage computation by aggregating indistinguishable executions.
- **Illustrative end-to-end example:** We provide a detailed example demonstrating the full pipeline from DT modeling to leakage quantification, showcasing the framework's applicability and effectiveness.

Paper Outline. The rest of the paper is organized as follows. Section II proposes a process-algebraic model for DTs with labeled transition semantics. In Section III, we review the preliminary concepts in quantifying information leakage, including MDPs, MCs, schedulers, and resolving nondeterminism. Section IV specifies the semantics of DTs as Markovian processes, which sets the stage to analyze them from attacker's viewpoint and measure the leakage. In addition, it includes a step-by-step illustrative example of quantifying leakage in a DT. In Section V, we discuss the limitations of the work. Related work is specified in Section VI. Finally, Section VII concludes the paper and specifies future work briefly.

II. Hybrid Calculus of Digital Twins

A high-level view of CPSs consists of four major components: the physical plant, cyber component, sensors, and actuators. Sensors measure physical values from the plant and pass them to the cyber component. The cyber component adjusts the actuators in order to give inputs to the plant. Actuators, sensors and the physical plant constitute the environment, whereas the cyber component makes the controller of the CPS. Figure 1 demonstrates these major components.

A. Environments and Processes

Inspired by CCPS [12], we define the calculus of DTs in this section by restricting the values to floating point numbers in both cyber and physical environments. Let's denote the set of floating point numbers of size s with $\mathbb{R}_\mathsf{s}$.

a) Preliminary Notations: Let $x, h, l \in \mathcal{X}$ be the state variables, $c \in \mathcal{C}$ range over communication channels, $a \in \mathcal{A}$ range over actuator names, $s \in \mathcal{S}$ range over sensor device names, and $v \in \mathcal{V}$ range over values which can be booleans,

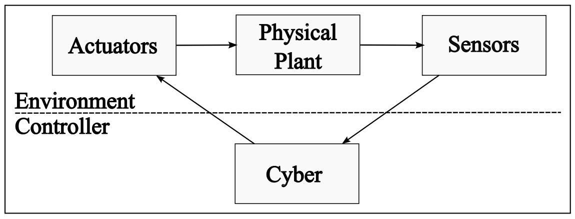

Fig. 1: Major components of a CPS.

integers, and floating point numbers of certain size s. Let $\mathcal{N}$ be a generic set of names. Then, $\mathbb{R}_\mathsf{s}^{\mathcal{N}} = \{\xi \mid \xi : \mathcal{N} \to \mathbb{R}_\mathsf{s}\}$ is the set of all functions from $\mathcal{N}$ to $\mathbb{R}_\mathsf{s}$. $\xi[n \mapsto v](x)$ is a function that maps n to v and acts identical to ξ for any other input. We denote $\xi \leq \xi'$ if for all $x \in \mathcal{N}$, $\xi(x) \leq \xi'(x)$.

A DT $E \bowtie P$ consists of physical environment E and process (cyber) P, defined in Definitions II.1 and II.2, respectively.

Definition II.1 (Physical environment: E)**.** *Let $\hat{\mathcal{X}} \subseteq \mathcal{X}$, $\hat{\mathcal{A}} \subseteq \mathcal{A}$, and $\hat{\mathcal{S}} \subseteq \mathcal{S}$. Then, the physical environment E is defined as $(\xi_\mathrm{x}, \xi_\mathrm{u}, \xi_\mathrm{w}, evol, \xi_\mathrm{e}, meas, inv)$, where: (i) $\xi_\mathrm{x} \in \mathbb{R}_\mathsf{s}^{\hat{\mathcal{X}}}$ is the state function, i.e., it maps state variables to floating point numbers. (ii) $\xi_\mathrm{u} \in \mathbb{R}_\mathsf{s}^{\hat{\mathcal{A}}}$ is the actuator function, i.e., it maps actuator names to floating point numbers. (iii) $\xi_\mathrm{w} \in \mathbb{R}_\mathsf{s}^{\hat{\mathcal{X}}}$ is the evolution uncertainty function, i.e., it maps a state variable to a floating point number denoting the maximum distance between its value and its representation in the model. (iv) $evol : \mathbb{R}_\mathsf{s}^{\hat{\mathcal{X}}} \times \mathbb{R}_\mathsf{s}^{\hat{\mathcal{A}}} \times \mathbb{R}_\mathsf{s}^{\hat{\mathcal{X}}} \to \mathcal{P}(\mathbb{R}_\mathsf{s}^{\hat{\mathcal{X}}})$ is the evolution map that receives a state function, an actuator function, and the uncertainty function and returns a (finite) set of next possible states. (v) $\xi_\mathrm{e} \in \mathbb{R}_\mathsf{s}^{\hat{\mathcal{S}}}$ is the sensor error function (measurement error function), i.e., it maps each sensor name to a floating point number that denotes the maximum distance between its measured value and its actual value in the model. (vi) $meas : \mathbb{R}_\mathsf{s}^{\hat{\mathcal{X}}} \times \mathbb{R}_\mathsf{s}^{\hat{\mathcal{S}}} \to \mathcal{P}(\mathbb{R}_\mathsf{s}^{\hat{\mathcal{S}}})$ is the measurement map, that receives a state function and an error function and returns a set of possible measurement functions. (vii) $inv : \mathbb{R}_\mathsf{s}^{\hat{\mathcal{X}}} \to \{\top, \bot\}$ is the invariable function. If $inv(\xi_\mathrm{x}) = \bot$, then the DT's E is in deadlock. If $inv(\xi_\mathrm{x}) = \top$ then the evolution of E can continue.*

We assume that evolution is monotonic wrt uncertainty, i.e., if $\xi_\mathrm{w} \leq \xi'_\mathrm{w}$ then $evol(\xi_\mathrm{x}, \xi_\mathrm{u}, \xi_\mathrm{w}) \subseteq evol(\xi_\mathrm{x}, \xi_\mathrm{u}, \xi'_\mathrm{w})$.

We define the cyber component (controller) using a timed process algebra.

Definition II.2 (Syntax of processes: P)**.** *The syntax of processes is defined as $P, Q ::= \mathsf{nil} \mid \mathsf{idle}.P \mid P \parallel Q \mid \lfloor \pi.P \rfloor Q \mid [b]\{P\}, \{Q\} \mid P \backslash c \mid X \mid \mathsf{rec}X.P$.*

nil is the terminal process. $\mathsf{idle}.P$ sleeps for one unit of time and then proceeds as P, and so P is time-guarded. $P \parallel Q$ is the parallel composition of P and Q. $\lfloor \pi.P \rfloor Q$ is the prefix with timeout, such that π ranges over communication prefixes $\mathsf{snd}\ c\langle v \rangle$, $\mathsf{rcv}\ c(x)$, $\mathsf{read}\ s(x)$, and $\mathsf{write}\ a\langle v \rangle$. These prefixes respectively represent sending value v on channel c, receiving a value on channel c and binding it to x, reading a value from sensor s, and writing value v to actuator a. $\lfloor \pi.P \rfloor Q$ does the communication π successfully within one time unit and then continues as P. Otherwise, i.e., if no communication partner exists within one time unit, it continues as Q. Q is time-guarded in this structure. $P\backslash c$ is the channel restriction. $[b]\{P\},\{Q\}$ is the conditional process where b is boolean. X is the process variable. $\text{rec}X.P$ is the time-guarded recursion, i.e., all occurrences of X in P must be time-guarded.

Note that in $\lfloor \mathsf{rcv}\ c(x).P \rfloor Q$ and $\lfloor \mathsf{read}\ s(x).P \rfloor Q$, x is bound, and X is bound in $\text{rec}X.P$. We identify processes up to α-conversion. We also identify DTs up to renaming state variable names, actuators, and sensors. One syntactic sugar that we will use in the rest of the paper is defined as $\pi.P \triangleq \text{rec}X.\lfloor \pi.P \rfloor X$, which intuitively requires the prefix π to be followed by P after potentially multiple time units.

Through a well-formedness property, given below, we ensure that sensors and actuators expressed in processes have already been defined in the environment.

Definition II.3 (Well-formedness of DTs). *DT $E \bowtie P$ is well-formed iff (i) for any sensor s in P, $\xi_{\mathrm{e}}(s)$ is defined in E, and (ii) for any actuator a in P, $\xi_{\mathrm{u}}(a)$ is defined in E.*

B. Labeled Transition System of Digital Twins

In order to define the labeled transition system (LTS) of DTs, we first define some auxiliary functions about the environments to simplify the expression of LTS. Let $E = (\xi_{\mathrm{x}}, \xi_{\mathrm{u}}, \xi_{\mathrm{w}}, \mathit{evol}, \xi_{\mathrm{e}}, \mathit{meas}, \mathit{inv})$. We define $\mathit{read\_sensor}(E, s) = \{\xi(s) \mid \xi \in \mathit{meas}(\xi_{\mathrm{x}}, \xi_{\mathrm{e}})\}$, i.e., the set of all possible measurements of sensor s at state ξ_{x} with error ξ_{e}. Let $\mathit{update\_act}(E, a, v) = (\xi_{\mathrm{x}}, \xi_{\mathrm{u}}[a \mapsto v], \xi_{\mathrm{w}}, \mathit{evol}, \xi_{\mathrm{e}}, \mathit{meas}, \mathit{inv})$, i.e., the actuator function is updated in E. The set of all possible environments by evolving state in E is defined as $\mathit{next}(E) = \{(\xi, \xi_{\mathrm{u}}, \xi_{\mathrm{w}}, \mathit{evol}, \xi_{\mathrm{e}}, \mathit{meas}, \mathit{inv}) \mid \xi \in \mathit{evol}(\xi_{\mathrm{x}}, \xi_{\mathrm{u}}, \xi_{\mathrm{w}})\}$. Finally, we abuse inv notation and lift it to environments, i.e., $\mathit{inv}(E) = \mathit{inv}(\xi_{\mathrm{x}})$.

We first define the LTS of processes, which is then used to define the LTS of DTs.

LTS of Processes: LTS of processes is similar to the one presented in CCPS, with a slight modification. We introduce a set $\mathcal{M}$ that contains all mappings from names to values (a memory). While this addition is not central to the specification of DT semantics, it becomes essential when studying information flow. We assume that all bound variables in the memory are fresh through α-conversion. Let $\mathcal{M}$ be a set of bindings (pairs) of names and values. The configurations are then pairs of $P, \mathcal{M}$. Figure 2 defines the LTS of processes, in which $\lambda \in \{\mathsf{idle}, \tau, \bar{c}v, cv, a!v, s?v\}$ ranges over the transition labels. Majority of these rules align with standard process algebra for timed systems. Specifically, the 6th rule propagates untimed actions over parallel composition, while the 12th rule enforces communication whenever it is possible, as defined by rule 5.

LTS of DTs: LTS of DTs is defined using the configuration $E \bowtie P, \mathcal{M}$. We are defining $E \bowtie P, \mathcal{M} \xrightarrow{\alpha} E' \bowtie P', \mathcal{M}'$ in Figure 3, where $\alpha \in \{\mathsf{idle}, \tau, cv, \bar{c}v\}$ ranges over the possible

1
$$\lfloor \mathsf{snd}\ c\langle v\rangle.P \rfloor Q, \mathcal{M} \xrightarrow{\bar{c}v} P, \mathcal{M}$$

2
$$\lfloor \mathsf{rcv}\ c(x).P \rfloor Q, \mathcal{M} \xrightarrow{cv} P\{v/x\}, \mathcal{M} \cup \{(x, v)\}$$

3
$$\lfloor \mathsf{write}\ a\langle v\rangle.P \rfloor Q, \mathcal{M} \xrightarrow{a!v} P, \mathcal{M}$$

4
$$\lfloor \mathsf{read}\ s(x).P \rfloor Q, \mathcal{M} \xrightarrow{s?v} P\{v/x\}, \mathcal{M} \cup \{(x, v)\}$$

5
$$\frac{P, \mathcal{M}_1 \xrightarrow{\bar{c}v} P', \mathcal{M}_1' \qquad Q, \mathcal{M}_2 \xrightarrow{cv} Q', \mathcal{M}_2'}{P \parallel Q, \mathcal{M}_1 \cup \mathcal{M}_2 \xrightarrow{\tau} P' \parallel Q', \mathcal{M}_1' \cup \mathcal{M}_2'}$$

6
$$\frac{P, \mathcal{M} \xrightarrow{\lambda} P', \mathcal{M}' \qquad \lambda \neq \mathsf{idle}}{P \parallel Q, \mathcal{M} \cup \mathcal{M}_Q \xrightarrow{\lambda} P' \parallel Q, \mathcal{M}' \cup \mathcal{M}_Q}$$

7
$$\frac{P, \mathcal{M} \xrightarrow{\lambda} P', \mathcal{M}' \qquad \lambda \notin \{cv, \bar{c}v\}}{P\backslash c, \mathcal{M} \xrightarrow{\lambda} P'\backslash c, \mathcal{M}'}$$

8
$$\frac{P\{\text{rec}X.P/X\}, \mathcal{M} \xrightarrow{\lambda} Q, \mathcal{M}'}{\text{rec}X.P, \mathcal{M} \xrightarrow{\lambda} Q, \mathcal{M}'}$$

9
$$\mathsf{nil}, \mathcal{M} \xrightarrow{\mathsf{idle}} \mathsf{nil}, \mathcal{M}$$

10
$$\mathsf{idle}.P, \mathcal{M} \xrightarrow{\mathsf{idle}} P, \mathcal{M}$$

11
$$\lfloor \pi.P \rfloor Q, \mathcal{M} \xrightarrow{\mathsf{idle}} Q, \mathcal{M}$$

12
$$\frac{P, \mathcal{M}_1 \xrightarrow{\mathsf{idle}} P', \mathcal{M}_1' \qquad Q, \mathcal{M}_2 \xrightarrow{\mathsf{idle}} Q', \mathcal{M}_2' \qquad P \parallel Q, \mathcal{M}_1 \cup \mathcal{M}_2 \not\xrightarrow{\tau}}{P \parallel Q, \mathcal{M}_1 \cup \mathcal{M}_2 \xrightarrow{\mathsf{idle}} P' \parallel Q', \mathcal{M}_1' \cup \mathcal{M}_2'}$$

13
$$\frac{P, \mathcal{M} \xrightarrow{\lambda} P', \mathcal{M}' \qquad b \rightsquigarrow \mathit{true}}{[b]\{P\},\{Q\}, \mathcal{M} \xrightarrow{\lambda} P', \mathcal{M}'}$$

14
$$\frac{Q, \mathcal{M} \xrightarrow{\lambda} Q', \mathcal{M}' \qquad b \rightsquigarrow \mathit{false}}{[b]\{P\},\{Q\}, \mathcal{M} \xrightarrow{\lambda} Q', \mathcal{M}'}$$

Fig. 2: LTS of processes: $P, \mathcal{M} \xrightarrow{\lambda} P', \mathcal{M}'$.

activities of DT, comprising (i) the non-observable activity τ which includes reading from a sensor, writing into an actuator, and internal communication of (sub)processes, (ii) observable logical activities cv and $\bar{c}v$, i.e., channel transmissions with the processes external to the DT, and (iii) the time passage activity idle which evolves environment (rule 6). In all rules of Figure 3, $\mathit{inv}(E)$ is a precondition and removed for brevity.

Without modifying CCPS semantics, properties such as time wellness and bisimulation remain unchanged. However, we do not cover these aspects here as they are not central to the focus of QIF in this work.

Example II.4. *Consider a DT with floating-point precision fixed to one digit. DT's initial state $s_0 = E_0 \bowtie P_0, \mathcal{M}_0$ consists of:*

- *$E_0 = (\xi_{\mathrm{x}}, \xi_{\mathrm{u}}, \xi_{\mathrm{w}}, \mathit{evol}, \xi_{\mathrm{e}}, \mathit{meas}, \mathit{inv})$, where h and x are environmental vairables, and $\xi_{\mathrm{x}}(x) = 0.0$. The uncertainty of evolution is defined for x as $\xi_{\mathrm{w}}(x) = 0.1$. We have a single sensor s, with measurement error $\xi_{\mathrm{e}}(s) = 0.5$. We have a single actuator a, where $\xi_{\mathrm{u}}(a) = 0.2$. Evolution is defined as $\mathit{evol}(\xi_{\mathrm{x}}, \xi_{\mathrm{u}}, \xi_{\mathrm{w}})\{\xi \mid \xi(x) = \xi_{\mathrm{x}}(x) + \xi_{\mathrm{u}}(a) + \gamma, \gamma \in \{-0.1, 0.0, 0.1\}\}$. Sensor measurement is defined as $\mathit{meas}(\xi_{\mathrm{x}}, \xi_{\mathrm{e}}) = \{\xi \mid \xi(s) = \xi_{\mathrm{x}}(h) + \gamma, \gamma \in \{-0.5, -0.4, \ldots, 0.5\}\}$. Finally, $\mathit{inv}(\xi_{\mathrm{x}}) = (\xi_{\mathrm{x}}(x) < 0.1)$.*

1
$$\frac{P, \mathcal{M} \xrightarrow{\bar{c}v} P', \mathcal{M}'}{E \bowtie P, \mathcal{M} \xrightarrow{\bar{c}v} E \bowtie P', \mathcal{M}'}$$

2
$$\frac{P, \mathcal{M} \xrightarrow{cv} P', \mathcal{M}'}{E \bowtie P, \mathcal{M} \xrightarrow{cv} E \bowtie P', \mathcal{M}'}$$

3
$$\frac{P, \mathcal{M} \xrightarrow{s?v} P', \mathcal{M}' \qquad v \in read\_sensor(E, s)}{E \bowtie P, \mathcal{M} \xrightarrow{\tau} E \bowtie P', \mathcal{M}'}$$

4
$$\frac{P, \mathcal{M} \xrightarrow{a!v} P', \mathcal{M}' \qquad E' = update\_act(E, a, v)}{E \bowtie P, \mathcal{M} \xrightarrow{\tau} E' \bowtie P', \mathcal{M}'}$$

5
$$\frac{P, \mathcal{M} \xrightarrow{\tau} P', \mathcal{M}'}{E \bowtie P, \mathcal{M} \xrightarrow{\tau} E \bowtie P', \mathcal{M}'}$$

6
$$\frac{P, \mathcal{M} \xrightarrow{\mathsf{idle}} P', \mathcal{M}' \qquad E \bowtie P, \mathcal{M} \not\xrightarrow{\tau} \qquad E' \in next(E)}{E \bowtie P, \mathcal{M} \xrightarrow{\mathsf{idle}} E' \bowtie P', \mathcal{M}'}$$

Fig. 3: LTS of DTs: $E \bowtie P, \mathcal{M} \xrightarrow{\alpha} E' \bowtie P', \mathcal{M}'$.

- $P_0 =$ read $s(h).([h > 0.4]\{$snd $c\langle 1\rangle$.nil$\}\{$snd $c\langle 2\rangle$.nil$\} \parallel$ rcv $c(l)$.nil$)\backslash c$.
- $\mathcal{M}_0 = \{(l, 0.0)\}$.

The sequence of states in the path are generally as follows: First, the sensor reads h (with potential error), then based on the read value of h either 1 or 2 is assigned to l. Next, in timed evolution, the environment reaches to a deadlock state. For simplicity, we assume that both evolution uncertainty and sensor measurement error are modeled using a uniform distribution.

III. Quantitative Information Flow

In this section, we present the preliminary concepts and notations essential for the proposed approach to QIF. While most of this material is covered in detail by Salehi et al. [28], we include it here to provide a self-contained presentation of the formal framework.

A probability distribution is $Pr : \mathbb{X} \to [0,1]$ over set $\mathbb{X}$, where $\Sigma_{x\in\mathbb{X}} Pr(x) = 1$. $\mathcal{D}(\mathbb{X}) = \{Pr \mid Pr : \mathbb{X} \to [0,1]\}$ is the set of all distributions for set $\mathbb{X}$. Considering equivalence relation R on set S, R-equivalence classes and quotient spaces S/R are defined in the standard format.

Having X as a random variable over finite set $\mathbb{X}$, its vulnerability is $Vul(X) = \max_{x\in\mathbb{X}} Pr(X = x)$. Renyi's min entropy represents knowledge uncertainty about X and is defined according to its vulnerability: $H_\infty(\mathbb{X}) = -\log_2 Vul(X)$.

Next, we define Markov decision processes which will be used to model DTs for QIF.

Definition III.1 (Markov Decision Process (MDP)). *An MDP M is the tuple $(S, Act, \mathbf{P}, \zeta, AP, V)$ where: (i) S is the set of states. (ii) Act is the set of actions. (iii) $\mathbf{P} : S \to Act \to S \to [0,1]$ is the transition probability such that $\forall s \in S, \alpha \in Act, \Sigma_{s'\in S}\mathbf{P}(s)(\alpha)(s') \in \{0,1\}$, i.e., action α is either enabled in a state or not. (iv) $\zeta : S \to [0,1]$ is the initial state probability distribution. Therefore, $\Sigma_{s\in S}\zeta(s) = 1$. (v) AP is the set of atomic propositions. (vi) $V : S \to AP$ is the labeling function.*

We denote the set of all enabled actions in a state s as $Act(s)$. The set of all initial states of an MDP M is represented by $Init(M)$. An execution path σ is defined as a sequence of states connected by transitions. The notation $PathFrags(s, s')$ specifies the path fragments between a starting state s and an ending state s'. A trace is a sequence of labels (atomic propositions), and multiple execution paths can correspond to the same trace.

To ensure MDP M is non-blocking, a self-loop is added to each state s with no successors, defined as $\mathbf{P}(s)(\varepsilon)(s) = 1$. Here, the distinguished action label ε indicates that the self-loop's action is not of further significance. Such states are referred to as final states, and the set of all final states in M is denoted by $final(M)$.

Definition III.2 (Markov Chain (MC)). *A discrete time MC M is the tuple $(S, \mathbf{P}, \zeta, AP, V)$ where $\mathbf{P} : S \times S \to [0,1]$ is the transition probability between states, such that $\forall s \in S, \Sigma_{s'\in S}\mathbf{P}(s, s') = 1$. Other components of MC are defined identical to Definition III.1.*

The probability of following finite execution path σ, $Pr(\sigma = s_0 \cdots s_n)$, is defined as (i) $\zeta(s_0)$, if $n = 0$, and (ii) $\zeta(s_0) \cdot \Pi_{0\leq i\leq n-1} Pr(s_i, s_{i+1})$, if $n \neq 0$. Then, reachability probability of state s is $Pr(s) = \Sigma_{\sigma\in PathFrags(s_0,s)\wedge s_0\in Init(M)} Pr(\sigma)$.

We assume that DTs always terminate, with states representing the current values of variables. Consequently, Markovian models of terminating DTs take the form of a directed acyclic graph (DAG), except for self-loops in final states. In this structure, reachability probabilities align with long-run probabilities [30]. Initial states correspond to the roots of the DAG, while final states appear as leaves.

A probabilistic scheduler defines the scheduling policy of a DT by determining the order and probability of executing (sub)processes. When applied, it resolves nondeterministic choices in a DT by replacing them with probabilistic ones. Since we model concurrency in (sub)processes using nondeterminism in MDPs, the scheduler addresses this nondeterminism. For simplicity, we focus on a key subclass known as *memoryless probabilistic schedulers (MPSs)*. An MPS, given a state s, assigns a probability to each action $\alpha \in Act(s)$, with the random choice being independent of the execution history, hence the term *memoryless*.

Definition III.3 (Memoryless probabilistic scheduler (MPS)). *Let $M = (S, Act, \mathbf{P}, \zeta, AP, V)$ be an MDP. $\delta : S \to \mathcal{D}(Act)$ is an MPS such that $\delta(s) \in \mathcal{D}(Act(s))$ for all $s \in S$.*

Using MPS δ, MC M_δ is induced from MDP M, according to Definition III.4.

Definition III.4 (Induced MC of an MDP). *Let $M = (S, Act, \mathbf{P}, \zeta, AP, V)$ be an MDP and $\delta : S \to \mathcal{D}(Act)$ be an MPS. The MC of M induced by δ is $M_\delta = (S, \mathbf{P}_\delta, \zeta, AP, V)$, where $\mathbf{P}_\delta(s, s') = \Sigma_{\alpha\in Act(s)}\delta(s)(\alpha) \cdot \mathbf{P}(s)(\alpha)(s')$*

IV. Leakage Quantification in Digital Twins

In this section, we define the semantics of DTs using Markovian models by specifying the MDP representation of a DT $E \bowtie P$. Let $M^{E \bowtie P}$ be the tuple $(S, Act, \mathbf{P}, \zeta, AP, V)$, where

- The set S represents the LTS configurations of DTs, where each $s \in S$ takes the form $E \bowtie P, \mathcal{M}$. We assume the existence of specific variables $l, h \in \mathcal{X}$, whose bound values can be stored in $\mathcal{M}$. Here, l and h represent public and secret data, respectively, from the attacker's perspective. For scenarios with multiple public or secret data components, l and h are treated as records (concatenations) of such data. Let Val_l and Val_h denote the domains of possible values for l and h, respectively.
- The set of activities in the LTS along with ε define $Act = \{\varepsilon, \tau, \mathsf{idle}, cv, \bar{c}v\}$. It is important to note that, from a given state, an action may lead to transitions into multiple states in a nondeterministic manner. This is consistent with the LTS of DTs, as multiple evaluation contexts may lead to the same activity in a process.
- $\mathbf{P}$ represents the probability of transitioning between states via specific actions in the DT. Evolution uncertainty ξ_{w} and sensor measurement errors ξ_{e} introduce probabilistic transitions between states under the same action. Sensor measurement errors are commonly modeled using different probability distributions, where the mean represents the true value and the variance accounts for measurement noise [31]. Regarding the uncertainty in system evolution, probabilistic modeling is a standard method to capture the range and likelihood of different outcomes [32]. This probabilistic modeling is distinct from nondeterministic choices, which pertain to selecting among different (sub)processes within the system to execute.
- $\zeta(s) = Pr(h = \bar{h})$, where $s = (E \bowtie P, \mathcal{M})$, $E = (\xi_{\mathrm{x}}, \cdots)$, and $\xi_{\mathrm{x}}(h) = \bar{h}$.
- $AP = Val_l$, i.e., the domain of values for l.
- $V(E \bowtie P, \mathcal{M}) \in Val_l$, i.e., V maps a state to a value of l as its label.

Nondeterminism in MDP ${}^{E \bowtie P}$ is resolved using MPS δ (Definition III.3), and thus MC $M_{\delta}^{E \bowtie P}$ is induced according to Definition III.4.

Example IV.1. *Consider the DT in Example II.4. Assume h is either 0.0 or 1.0 and remains unchanged (as confidentiality is the focus). We will consider these two cases in the following.*

Case 1: *First, let's assume $\xi_{\mathrm{x}}(h) = 0.0$.*

- *From s_0 eleven states $s_{1,1}, s_{1,2}, \ldots, s_{1,11}$ are accessible, each with action τ and probability of transition $1/11$ such that for $i \in \{1, 2, \ldots, 11\}$, $s_{1,i} = E_0 \bowtie P_1, \mathcal{M}_{1,i}$, where $P_1 = ([h > 0.4]\{\mathsf{snd}\ c\langle 1\rangle.\mathsf{nil}\}\{\mathsf{snd}\ c\langle 2\rangle.\mathsf{nil}\} \parallel \mathsf{rcv}\ c(l).\mathsf{nil})\backslash c$, and $\mathcal{M}_{1,1} = \{(h, -0.5)\}, \mathcal{M}_{1,2} = \{(h, -0.4)\}, \ldots, \mathcal{M}_{1,11} = \{(h, 0.5)\}$.*
- *From each $s_{1,i}$ ($i \in \{1, 2, \ldots, 11\}$), a single state $s_{2,i}$ is accessible with action τ and probability 1, such that $s_{2,i} = E_0 \bowtie P_2, \mathcal{M}_{2,i}$, where $P_2 = (\mathsf{nil} \parallel \mathsf{nil})\backslash c$, $\mathcal{M}_{2,i} = \mathcal{M}_{1,i} \cup \{(l, 2)\}$ for $i \in \{1, 2, \ldots, 10\}$, and $\mathcal{M}_{2,11} = \mathcal{M}_{1,11} \cup \{(l, 1)\} = \{(h, 0.5), (l, 1)\}$.*
- *From each $s_{2,i}$ ($i \in \{1, 2, \ldots, 11\}$) three distinct states $s_{3,3i-2} = (E_{3,3i-2} \bowtie P_2, \mathcal{M}_{2,i})$, $s_{3,3i-1} = (E_{3,3i-1} \bowtie P_2, \mathcal{M}_{2,i})$ and $s_{3,3i} = (E_{3,3i} \bowtie P_2, \mathcal{M}_{2,i})$ are accessible with action* idle *and probability $1/3$ such that for (i) $E_{3,3i-2}$, $\xi_{\mathrm{x}}(x) = 0.0 + 0.2 - 0.1 = 0.1$, (ii) $E_{3,3i-1}$, $\xi_{\mathrm{x}}(x) = 0.0 + 0.2 + 0.0 = 0.2$, and (iii) $E_{3,3i}$, $\xi(x) = 0.0 + 0.2 + 0.1 = 0.3$. Note that $inv(E_{3,j}) = \bot$ for all $j \in \{1, \ldots, 33\}$.*

The left DAG in Figure 4 demonstrates the resulting MDP[1]*. The DAG consists of 56 states and 33 different paths from initial to terminal states. States s_0 and $s_{1,i}$, for $i \in \{1, \ldots, 11\}$, have label $l = 0$, whereas states $s_{2,i}$, for $i \in \{1, \ldots, 10\}$, have label $l = 2$, and $s_{2,11}$ has label $l = 1$. States $s_{3,i}$, for $i \in \{1, \ldots, 30\}$, have label $l = 2$, and states $s_{3,i}$ for $i \in \{31, 32, 33\}$ have label $l = 1$.*

Case 2: *Alternatively, if $\xi_{\mathrm{x}}(h) = 1.0$ initially, the DAG structure will be the same as Case 1, given as the middle DAG in Figure 4, with the following differences in state details.*

- *$\mathcal{M}'_{1,1} = \{(h, 0.5)\}, \mathcal{M}'_{1,2} = \{(h, 0.6)\}, \ldots, \mathcal{M}'_{1,11} = \{(h, 1.5)\}$,*
- *$\mathcal{M}'_{2,i} = \mathcal{M}'_{1,i} \cup \{(l, 1)\}$ for all $i \in \{1, \ldots, 11\}$,*
- *States s'_0 and $s'_{1,i}$ ($i \in \{1, \ldots, 11\}$) have label $l = 0$, and states $s'_{2,i}$ ($i \in \{1, \ldots, 11\}$) and $s'_{3,i}$ ($i \in \{1, \ldots, 33\}$) have label $l = 1$.*

In this example, each state has only one possible action to transition to the next state, so the choice of scheduler does not impact the structure of MC DAGs, i.e., $M_{\delta}^{E \bowtie P}$ is structurally identical to $M^{E \bowtie P}$. The only difference is that each transition in $M_{\delta}^{E \bowtie P}$ is associated solely with its transition probability.

A. Attacker's View of the DT

To quantify the information an attacker can infer about h, we construct the attacker's view of the DT. The attacker can distinguish final states from non-final ones by observing execution termination but cannot differentiate between paths with identical traces. For example, in the induced MC in Example IV.1, the attacker observes traces $0, 0, 1, 1^{\epsilon}$ and $0, 0, 2, 2^{\epsilon}$, while there are 66 distinct execution paths. Consequently, the attacker cannot distinguish final states sharing the same public values and trace. Additionally, while secret values in final states remain unknown, the attacker can guess h using a probability distribution derived from possible values of h in each final state. These requirements are formalized through the back-bisimulation equivalence relation [28], denoted by $\sim_b$.

Definition IV.2. *Let $M_{\delta}^{E \bowtie P}$ be an MC. A back-bisimulation for $M_{\delta}^{E \bowtie P}$ is a binary relation R on S such that for all $(s_1, s_2) \in R$: (i) $V(s_1) = V(s_2)$, (ii) every predecessor s'_1 of s_1 has a corresponding predecessor s'_2 of s_2 with $(s'_1, s'_2) \in R$, and (iii) every predecessor s'_2 of s_2 has a corresponding predecessor s'_1 of s_1 with $(s'_1, s'_2) \in R$.*

[1]The labels of the leaves are identical to those of their parent states but are omitted from the figure for brevity. Additionally, the transition label idle is replaced with i in this figure where necessary to reduce visual clutter.

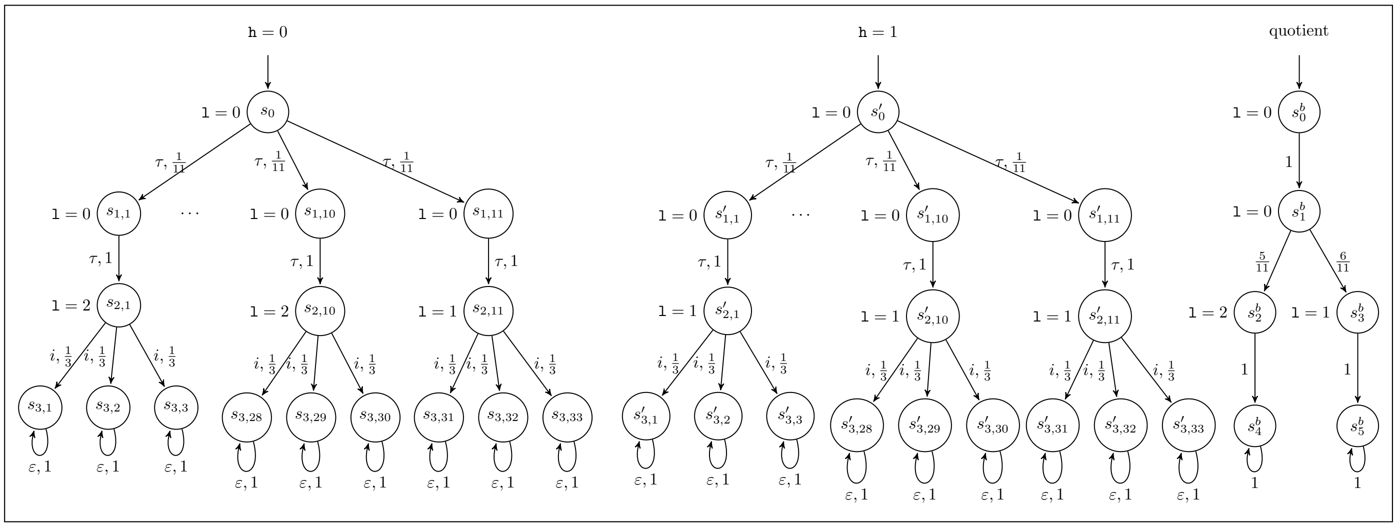

Fig. 4: Markovian model of example DT. From left to right: i) MDP in case $h = 0.0$, ii) MDP in case $h = 1.0$, and iii) back-bisimulation quotient (attacker's view).

States s_1 and s_2 are back-bisimilar, denoted $s_1 \sim_b s_2$, if a back-bisimulation R exists with $(s_1, s_2) \in R$.

In Definition IV.2, Condition (i) ensures that states s_1 and s_2 share the same public values. Condition (ii) requires every incoming transition of s_1 to have a matching incoming transition for s_2, while condition (iii) guarantees the reverse. It is straightforward to verify that $\sim_b$ is an equivalence relation, which partitions the state space of the MC into equivalence classes. The back-bisimulation quotient of $M_\delta^{E\bowtie P}$, denoted by $M_\delta^{E\bowtie P}/\sim_b$, is then defined accordingly.

Definition IV.3. *For MC $M_\delta^{E\bowtie P} = (S, \mathbf{P}_\delta, \zeta, Val_l, V)$ and relation $\sim_b$, the back-bisimulation quotient is defined as $M_\delta^{E\bowtie P}/\sim_b = (S/\sim_b, \mathbf{P}'_\delta, s^b_0, Val_l, V, Pr(h))$ where:*

- *$S/\sim_b$ is the quotient space of S under $\sim_b$,*
- *$\mathbf{P}'_\delta : (S/\sim_b) \times (S/\sim_b) \to [0,1]$ such that for all s^b and t^b, $\mathbf{P}'_\delta(s^b, t^b) = (\Sigma_{s\in s^b, t\in t^b} Pr(s) \cdot \mathbf{P}_\delta(s,t))/Pr(s^b)$, where $Pr(s)$ and $Pr(s^b)$ are the probabilities of reaching to s and s^b in $M_\delta^{E\bowtie P}$ and $M_\delta^{E\bowtie P}/\sim_b$ respectively,*
- *$s^b_0 = Init(M)$,*
- *$V([s]_{\sim_b}) = V(s)$, and*
- *$Pr(h)$ maps s^b to the probability distribution of h in s^b, $Pr(h_{s^b})$, such that $Pr(h_{s^b} = \bar{h}) = (\Sigma_{s\in s^b, s=(E\bowtie P, \mathcal{M}), \mathcal{M}=(\xi_x, \cdots), \xi_x(h)=\bar{h}} Pr(s))/Pr(s^b)$.*

Example IV.4. *Having defined the induced MC $M_\delta^{E\bowtie P}$ of the DT in Example IV.1, the back-bisimulation quotient $M_\delta^{E\bowtie P}/\sim_b$ is given as the right DAG in Figure 4. The label for s^b_0 and s^b_1 is $l = 0$, for s^b_2 and s^b_4 is $l = 1$, and for s^b_3 and s^b_5 is $l = 2$. The probability of transition from s^b_1 to s^b_2 is $5/11$, and to s^b_3 is $6/11$. Each state is an equivalence class, containing back-bisimilar states of $M_\delta^{E\bowtie P}$, given in the following: (i) $s^b_0 = \{s_0, s'_0\}$, (ii) $s^b_1 = \{s_{1,1}, s_{1,2}, \ldots, s_{1,11}, s'_{1,1}, s'_{1,2}, \ldots, s'_{1,11}\}$, (iii) $s^b_2 = \{s_{2,1}, s_{2,2}, \ldots, s_{2,10}\}$, (iv) $s^b_3 = \{s'_{2,1}, s'_{2,2}, \ldots, s'_{2,11}, s_{2,11}\}$, (v) $s^b_4 = \{s_{3,1}, s_{3,2}, \ldots, s_{3,30}\}$, and (vi) $s^b_5 = \{s'_{3,1}, s'_{3,2}, \ldots, s'_{3,33}, s_{3,31}, s_{3,32}, s_{3,33}\}$.*

$Pr(h)$ for each state is defined as follows: (i) $Pr(h_{s^b_0}) = Pr(h_{s^b_1}) = \{0 \mapsto 1/2, 1 \mapsto 1/2\}$, (ii) $Pr(h_{s^b_2}) = Pr(h_{s^b_4}) = \{0 \mapsto 1, 1 \mapsto 0\}$, and (iii) $Pr(h_{s^b_3}) = Pr(h_{s^b_5}) = \{0 \mapsto 1/12, 1 \mapsto 11/12\}$.

B. Leakage Quantification

In each state of $M_\delta^{E\bowtie P}/\sim_b$, $Pr(h)$ captures the attacker's uncertainty about h. The scheduler's chosen (sub)processes and the attacker's observed public values cause $Pr(h)$ to evolve along each trace of $M_\delta^{E\bowtie P}/\sim_b$, transforming the initial distribution of h into posterior distributions in the final states, denoted by $final(M_\delta^{E\bowtie P}/\sim_b)$.

The uncertainty about h in a state s^b, given $Pr(h_{s^b})$ is measured by $H_\infty(h_{s^b})$. The initial uncertainty is $H_\infty(h_{s^b_0})$, and the remaining uncertainty in the final states is the expected uncertainty:$\Sigma_{s^b_f \in final(M_\delta^{E\bowtie P}/\sim_b)} Pr(s^b_f) \cdot H_\infty(h_{s^b_f})$. The leakage of the DT $E \bowtie P$ running under the scheduler δ is then calculated as $\mathcal{L}(E \bowtie P) = H_\infty(h_{s^b_0}) - \Sigma_{s^b_f \in final(M_\delta^{E\bowtie P}/\sim_b)} Pr(s^b_f).H_\infty(h_{s^b_f})$.

Example IV.5. *In the DT of Example IV.1 with the back-bisimulation quotient in Figure 4, the leaked information amount is computed as 0.93 bits:*

$$\begin{aligned}
&\mathcal{L}(E \bowtie P) = \\
&\quad H_\infty(h_{s^b_0}) - (Pr(s^b_4) \cdot H_\infty(h_{s^b_4}) + Pr(s^b_5) \cdot H_\infty(h_{s^b_5})) \\
&\quad = 1 - (5/11 \cdot 0 + 6/11 \cdot (-\log_2(11/12))) = 0.93.
\end{aligned}$$

V. Discussion

The proposed framework demonstrates significant potential for studying information leakage in CPSs by modeling their DTs using the calculus introduced in this paper. By formalizing information flow and quantifying leakage through the attacker's view, this method provides a structured approach to analyze confidentiality risks in CPSs. This approach is particularly valuable for understanding how public and private data interact within a system's operational processes, thereby enabling a deeper analysis of CPS behavior and potential vulnerabilities. Furthermore, the methodology aligns well with QIF theory, leveraging well-established principles to assess attacker knowledge.

Despite its strengths, the framework has limitations that warrant discussion. While the proposed calculus for DTs supports infinite execution paths, the current limitation stems from the requirements of QIF modeling, which necessitates finite traces to evaluate and compare the attacker's uncertainty before and after execution. To accommodate this, we restrict our analysis to finite execution paths. An alternative approach would be to truncate infinite paths at a predetermined length, enabling bounded analysis without modifying the leakage computation framework. This restriction aligns with the broader notion of approximation adopted in this work, where continuous CPS behaviors are discretized—both in value (e.g., floating-point representation) and in time (finite trace length)—to enable tractable QIF analysis.

Another limitation stems from the use of back-bisimulation to construct the attacker's view of the MC. As highlighted in previous work [28], back-bisimulation does not capture certain relationships among states that are indistinguishable to the attacker. Specifically, pseudoback-bisimilar states do not belong to the same back-bisimulation equivalence class, meaning that back-bisimulation fails to aggregate all paths with identical traces. Consequently, in scenarios involving pseudoback-bisimilar states, the MC constructed using back-bisimulation may incorrectly distinguish secret values in the final states of paths that the attacker cannot truly differentiate. For such cases, the trace-exploration-based methods [33] offers an alternative approach, that need further study in the context of CPSs.

VI. Related Work

a) Qualitative information flow in CPSs.: Several works have explored qualitative information flow in CPSs. For indirect information flow, Zhang et al. [34] introduce a semantics-based noninterference assessment method, employing both coarse and fine evaluations to determine event-aware security in CPSs. This approach enhances the precision of security assessments by considering the semantics of events within the system. Akella et al. [35] used process algebra to specify a semantic model for CPS information flows, though their approach omits the physical plant. Akella et al. [36] formalized confidentiality in the FREEDM smart grid [37] using process algebra to define nondeducibility and noninterference properties. Gamage et al. [38] employed finite state machines to analyze CPS information flow, while Lanotte et al. [39] proposed CCPSA, a process calculus for modeling indirect flows. Castellanos et al. [40] used data flow graphs and a reachability algorithm to analyze PLC programs, and Morris et al. [41] applied information-theoretic models to quantify CPS susceptibility to integrity attacks. Liu et al. [42] introduced a security architecture for CPSs that integrates input analysis, information flow analysis, and hardware verification. Direct information flow has been analyzed through taint tracking, both statically and dynamically in CPSs. Amir-Mohammadian [27] proposed a semantic framework to study direct information flows in CPSs. Liu et al. [43] conducted a case study on multi-stage manufacturing systems, extending dynamic taint analysis with manufacturing-specific taint propagation rules to enhance intrusion diagnosis. Maragkou et al. [44] surveyed information flow tracking methods aimed at protecting CPS against hardware Trojans, highlighting the potential of these methods in detecting such threats.

b) Quantitative information flow in CPSs.: Huang et al. [45] applied a variant of temporal logic to formalize DTs, providing formal statistical guarantees on information leakage during communication between the physical system and its digital counterpart. Feng et al. [46] proposed a framework that utilizes algorithmic quantification and statistical analysis to measure information leakage in CPSs, with a specific focus on smart grid applications. In addition, Feng et al. [47] examined a smart grid CPS from an information flow perspective, highlighting how physical observations can inadvertently lead to cyber information leakage. They proposed a method to quantify this flow. Zhang et al. [48] introduced a side-channel analysis-based model extraction attack targeting intelligent CPS. They developed an efficient framework that leverages side-channel information to reconstruct models within CPS.

VII. Conclusion and Future Work

We have introduced a framework for analyzing information leakage in CPSs by leveraging the modeling capabilities of DTs within a formal process-algebraic calculus. This calculus, equipped with LTS semantics, digitizes the behavior of physical environments into a finite domain, ensuring a finite number of successor states when modeling DTs as MDPs. By defining DT semantics through MDPs, which induce MCs under probabilistic schedulers, the framework effectively captures the nondeterministic and probabilistic evolution inherent to DTs and CPSs—stemming from system uncertainties, sensor errors, and the concurrent logic of cyber components. This formalization enables the precise quantification of information leakage by systematically evaluating an attacker's knowledge of DT secrets before and after execution based on observed public outputs, providing a structured and rigorous methodology for assessing confidentiality risks in CPSs. To illustrate the applicability of the framework, we presented a detailed example with a quantified leakage value that is both nontrivial and fractional, demonstrating the capability of the model to capture fine-grained confidentiality losses with decimal precision.

Future work includes extending our framework to quantify information leakage through side channels using process algebraic modeling. Another direction is leveraging process algebraic verification tools to analyze leakage in various DT specifications. Additionally, conducting case studies on real-world industrial DTs would enhance the framework's applicability and validation.

References

[1] J. Shi, J. Wan, H. Yan, and H. Suo, "A survey of cyber-physical systems," in *WCSP*. IEEE, 2011, pp. 1–6.

[2] P. Sharma and J. Gillanders, "Cybersecurity and forensics in connected autonomous vehicles: A review of the state-of-the-art," *IEEE Access*, vol. 10, pp. 108 979–108 996, 2022.

[3] K. Zhang, Y. Shi, S. Karnouskos, T. Sauter, H. Fang, and A. W. Colombo, "Advancements in industrial cyber-physical systems: An overview and perspectives," *IEEE Trans. Industr. Inform.*, vol. 19, no. 1, pp. 716–729, 2022.

[4] L. M. Oliveira, R. Dias, C. M. Rebello, M. A. Martins, A. E. Rodrigues, A. M. Ribeiro, and I. B. Nogueira, "Artificial intelligence and cyber-physical systems: A review and perspectives for the future in the chemical industry," *AI*, vol. 2, no. 3, p. 27, 2021.

[5] Q. Chen, H. Zhu, L. Yang, X. Chen, S. Pollin, and E. Vinogradov, "Edge computing assisted autonomous flight for UAV: Synergies between vision and communications," *IEEE Comm. Mag.*, vol. 59, no. 1, pp. 28–33, 2021.

[6] M. Achouch, M. Dimitrova, K. Ziane, S. Sattarpanah Karganroudi, R. Dhouib, H. Ibrahim, and M. Adda, "On predictive maintenance in industry 4.0: Overview, models, and challenges," *Applied Sciences*, vol. 12, no. 16, p. 8081, 2022.

[7] N. Aftabi, D. Li, T. Sharkey *et al.*, "An integrated cyber-physical risk assessment framework for worst-case attacks in industrial control systems," *arXiv preprint arXiv:2304.07363*, 2023.

[8] J. Yuan, J. Zhang, P. Qiu, X. Wei, and D. Liu, "A survey of of side-channel attacks and mitigation for processor interconnects," *Applied Sciences*, vol. 14, no. 15, p. 6699, 2024.

[9] T. A. Henzinger, "The theory of hybrid automata," in *IEEE LICS*. IEEE, 1996, pp. 278–292.

[10] A. Platzer, *Logical foundations of cyber-physical systems*. Springer, 2018, vol. 662.

[11] P. J. L. Cuijpers and M. A. Reniers, "Hybrid process algebra," *J. Log. Algebr. Program.*, vol. 62, no. 2, pp. 191–245, 2005.

[12] R. Lanotte and M. Merro, "A calculus of cyber-physical systems," in *LATA*. Springer, 2017, pp. 115–127.

[13] M. Hennessy and T. Regan, "A process algebra for timed systems," *Information and computation*, vol. 117, no. 2, pp. 221–239, 1995.

[14] F. Tao, Q. Qi, L. Wang, and A. Nee, "Digital twins and cyber–physical systems toward smart manufacturing and industry 4.0: correlation and comparison. engineering, 5 (4), 653–661," 2019.

[15] S. Camarella, M. P. Conway, K. Goering, and M. Huntington, "Digital twins: The next frontier of factory optimization," https://rb.gy/iycm09, 2024, accessed on 2025-02-05.

[16] D. Allison, P. Smith, and K. Mclaughlin, "Digital twin-enhanced incident response for cyber-physical systems," in *ARES*, 2023, pp. 1–10.

[17] "How using a digital twin will revolutionize your security," https://rb.gy/8g8c39, 2024, accessed on 2025-02-05.

[18] M. Eckhart and A. Ekelhart, "Digital twins for cyber-physical systems security: State of the art and outlook," *Security and Quality in Cyber-Physical Systems Engineering*, pp. 383–412, 2019.

[19] M. S. Alvim, K. Chatzikokolakis, A. McIver, C. Morgan, C. Palamidessi, and G. Smith, *The Science of Quantitative Information Flow*. Springer, 2020.

[20] L. M. Reimann, A. Prashar, C. Ghinami, R. Pelke, D. Sisejkovic, F. Merchant, and R. Leupers, "Qtflow: Quantitative timing-sensitive information flow for security-aware hardware design on RTL," in *VLSI TSA*. IEEE, 2024, pp. 1–4.

[21] B. Mao, W. Hu, A. Althoff, J. Matai, J. Oberg, D. Mu, T. Sherwood, and R. Kastner, "Quantifying timing-based information flow in cryptographic hardware," in *ICCAD*. IEEE Press, 2015, p. 552–559.

[22] N. Fernandes, A. McIver, and P. Sadeghi, "Explaining epsilon in local differential privacy through the lens of quantitative information flow," in *CSF*. IEEE, 2024, pp. 419–432.

[23] M. Jurado, R. G. Gonze, M. S. Alvim, and C. Palamidessi, "Analyzing the shuffle model through the lens of quantitative information flow," in *CSF*. IEEE, 2023, pp. 423–438.

[24] M. Jurado, C. Palamidessi, and G. Smith, "A formal information-theoretic leakage analysis of order-revealing encryption," in *CSF*. IEEE, 2021, pp. 1–16.

[25] G. Smith, "On the foundations of quantitative information flow," in *FoSSaCS*. Springer, 2009, pp. 288–302.

[26] F. Biondi, A. Legay, B. F. Nielsen, and A. Wasowski, "Maximizing entropy over markov processes," *JLAMP*, vol. 83, no. 5, pp. 384–399, 2014.

[27] S. Amir-Mohammadian, "A semantic framework for direct information flows in hybrid-dynamic systems," in *ACM CPSS*, 2021, pp. 5–15.

[28] K. Salehi, A. A. Noroozi, S. Amir-Mohammadian, and M. Mohagheghi, "An automated quantitative information flow analysis for concurrent programs," in *QEST*. Springer, 2022, pp. 43–63.

[29] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[30] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press Cambridge, 2008.

[31] P. Harris, P. F. Østergaard, S. Tabandeh, H. Söderblom, G. Kok, M. van Dijk, Y. Luo, J. Pearce, D. Tucker, A. P. Vedurmudi *et al.*, "Measurement uncertainty evaluation for sensor network metrology," *Metrology*, vol. 5, no. 1, p. 3, 2025.

[32] M. N. Asmat, S. U. R. Khan, and S. Hussain, "Uncertainty handling in cyber–physical systems: State-of-the-art approaches, tools, causes, and future directions," *J. Softw. Evol. Process.*, vol. 35, no. 7, p. e2428, 2023.

[33] A. A. Noroozi, J. Karimpour, and A. Isazadeh, "Information leakage of multi-threaded programs," *COMPUT ELECTR ENG.*, vol. 78, pp. 400–419, 2019.

[34] W. Zhong, J. Zhao, and H. Hu, "Semantics-based noninterference assessment in cyber-physical systems," *IEEE Trans. Comput. Soc. Syst.*, 2024.

[35] R. Akella, H. Tang, and B. M. McMillin, "Analysis of information flow security in cyber–physical systems," *Int. J. Crit. Infrastruct. Prot.*, vol. 3, no. 3-4, pp. 157–173, 2010.

[36] R. Akella and B. M. McMillin, "Information flow analysis of energy management in a smart grid," in *SAFECOMP*. Springer, 2010, pp. 263–276.

[37] A. Q. Huang, "Renewable energy system research and education at the nsf freedm systems center," in *IEEE PES GM*. IEEE, 2009, pp. 1–6.

[38] T. T. Gamage, B. M. McMillin, and T. P. Roth, "Enforcing information flow security properties in cyber-physical systems: A generalized framework based on compensation," in *IEEE COMPSAC*. IEEE, 2010, pp. 158–163.

[39] R. Lanotte, M. Merro, R. Muradore, and L. Viganò, "A formal approach to cyber-physical attacks," in *CSF*. IEEE, 2017, pp. 436–450.

[40] J. H. Castellanos, M. Ochoa, and J. Zhou, "Finding dependencies between cyber-physical domains for security testing of industrial control systems," in *ACSAC*, 2018, pp. 582–594.

[41] E. R. Morris, C. G. Murguia, and M. Ochoa, "Design-time quantification of integrity in cyber-physical systems," in *PLAS*, 2017, pp. 63–74.

[42] J. Liu, J. Corbett-Davies, A. Ferraiuolo, A. Ivanov, M. Luo, G. E. Suh, A. C. Myers, and M. Campbell, "Secure autonomous cyber-physical systems through verifiable information flow control," in *CPS-SPC*, 2018, pp. 48–59.

[43] T. Liu, B. Yang, Q. Li, J. Ye, W. Song, and P. Liu, "Cyber-physical taint analysis in multi-stage manufacturing systems (mms): A case study," *arXiv preprint arXiv:2109.12774*, 2021.

[44] S. Maragkou and A. Jantsch, "Information flow tracking methods for protecting cyber-physical systems against hardware trojans–a survey," *arXiv preprint arXiv:2301.02620*, 2022.

[45] L. Huang, L. R. Varshney, and K. E. Willcox, "Formal verification of digital twins with TLA and information leakage control," *arXiv preprint arXiv:2411.18798*, 2024.

[46] L. Feng and B. McMillin, "Information flow quantification framework for cyber physical system with constrained resources," in *COMPSAC*, vol. 2. IEEE, 2015, pp. 50–59.

[47] ——, "Quantification of information flow in a smart grid," in *COMPSAC*. IEEE, 2014, pp. 140–145.

[48] Q. Pan, J. Wu, X. Lin, and J. Li, "Side-channel analysis-based model extraction on intelligent cps: An information theory perspective," in *IEEE iThings and GreenCom and CPSCom and SmartData and Cybermatics*. IEEE, 2021, pp. 254–261.