Digital Twins

CPS and IoT Security

Alessandro Brighente

Master Degree in Cybersecurity

Università degli Studi di Padova

SPRITZ Security & Privacy Research Group

What is a Digital Twin

- A digital twin is a virtual model designed to accurately reflect a physical object
- “machines (physical and/or virtual) or computer-based models that are simulating, emulating, mirroring or twinning the life of a physical entity”
- We create a counterpart digital representation of an object to fully characterize its behavior
- The model we create is outfitted with various sensors related to vital areas of its functionality

Objective of a DT

- All these sensors produce data about the different aspects of the physical object
  - Energy output, temperature, weather conditions,...
- The data is relayed to a processing system and applied to the digital copy
- The model informed with such data can be used to run simulations, study performance issues, generate improvements, study cybersecurity aspects

DT or Simulations?

SPRITZ
Security & Privacy
Research Group

Università
degli Studi
di Padova

- Both DT and simulation use digital models to create a replica of a system's process, however
  - Simulations replicates a particular process
  - A DT replicates multiple processes and their interaction
- A DT thus provides way more features for the environment under study
- Furthermore, a DT usually operates on real-time data from sensors
- The DT virtual environment (virtual machines, containers, virtual networks) then creates a new flow of information by processing the real-time data

Spaces of a DT

- A DT comprises three main spaces
- **Physical space:** comprising real world devices such as sensors, actuators, and controllers (OT)
- **Digital space:** digital framework capable of simulating physical assets and their states, conditions, and configurations to make decisions on the physical space
- **Communication space:** creates the connection between the digital and the physical spaces

Digital Thread

- It is important to establish bidirectional interfaces in DTs
- *Digital thread* between physical space and digital space: the data from the physical space is processed by the digital space, which creates new useful information that can be sent back to the physical space
- Example: a DT synchronizes its models with respect to its physical counterpart to guarantee consistency in the production process (contextual parameters such as humidity, temperature, pressure)

Functional Layers

- We can abstract the functionalities of a DT by considering four layers
- **Layer 1: data dissemination and acquisition**, which captures the dynamics of the physical space and prepares instructions for physical assets
- **Layer 2: data management and synchronization**, which handles multi-source data to pass to layer 3, network management, and synchronization services

Functional Layers

- **Layer 3: data modeling and additional services**, which specifies digital models for shapes, behaviors, states and creates services such as monitoring, cybersecurity, and diagnostic
- **Layer 4: data visualization and accessibility,** which provides means for users, entities and processes to visualize the simulation results from the digital models and enable decisions on this basis
- The fundamental difference between layers is in the level of data they consider: from physical to digital model results

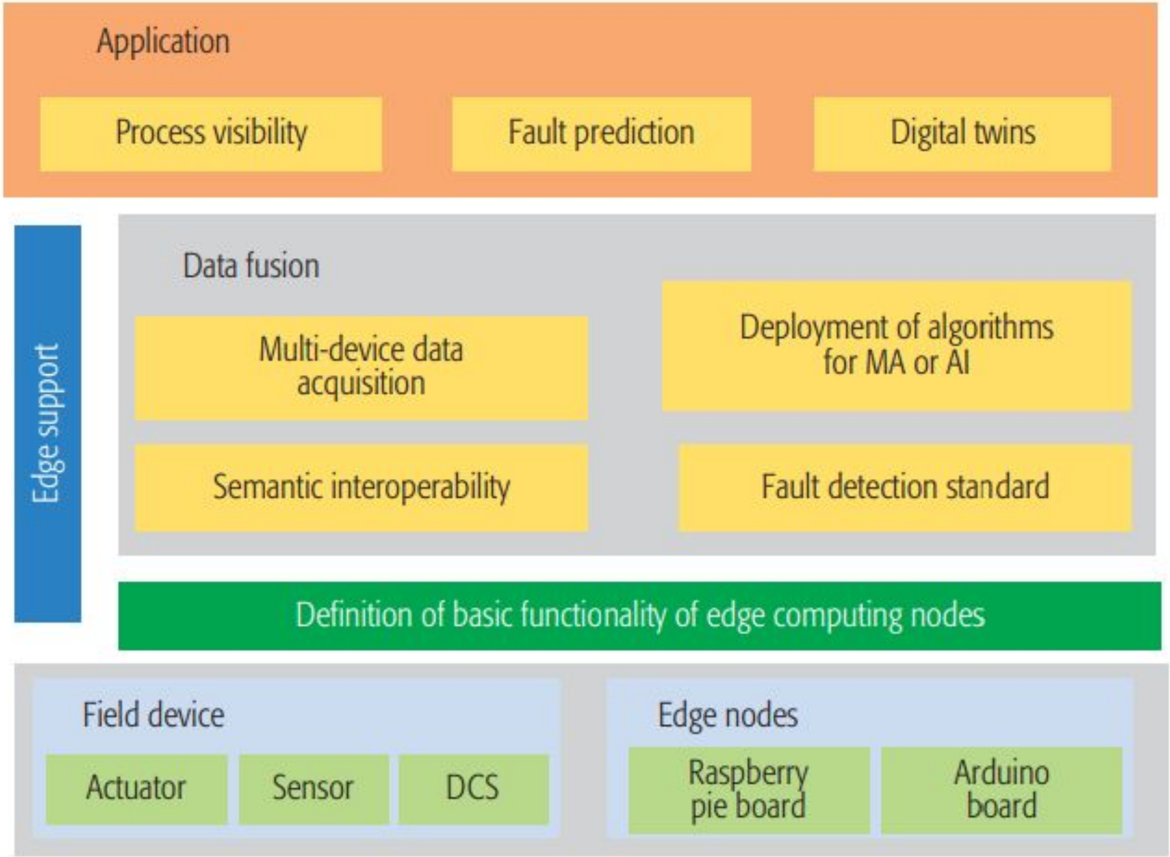
Keeping up

- A fundamental requirement of DTs is to keep the physical space and the digital space synchronized
- In case of a variation not recorded or inconsistent, the whole DT model may significantly deviate from the true world and can thus provide useless (or misleading) information
- Therefore, we should handle the complexities of the system in terms of infrastructure, communication, computation and storage

Infrastructure and Resource mgmt

- We need to take into account the currently available computing infrastructures to guarantee inter-layer synchronization
- We want to optimize the deployment of digital models and their processing during simulation
- Combined use of edge and fog computing: intensive operations can be executed by powerful devices, while less intensive tasks can be executed on resource limited edge devices

Infrastructure and Resource mgmt



Overhead Management

- We can identify three classes of overhead: communication, computation and storage
- Communication: layer 1 needs to consider many different protocols (TCP, Modbus, MQTT) which may however not be efficient
- Computation: volume of data produced by virtual assets, information collected from layer 1 elements, complexity of ML techniques (if any)
- Storage overhead: apply alternative techniques such as distributed file storage systems, non-relational database, redundant SQL servers,...

Use Case: Autonomous Driving

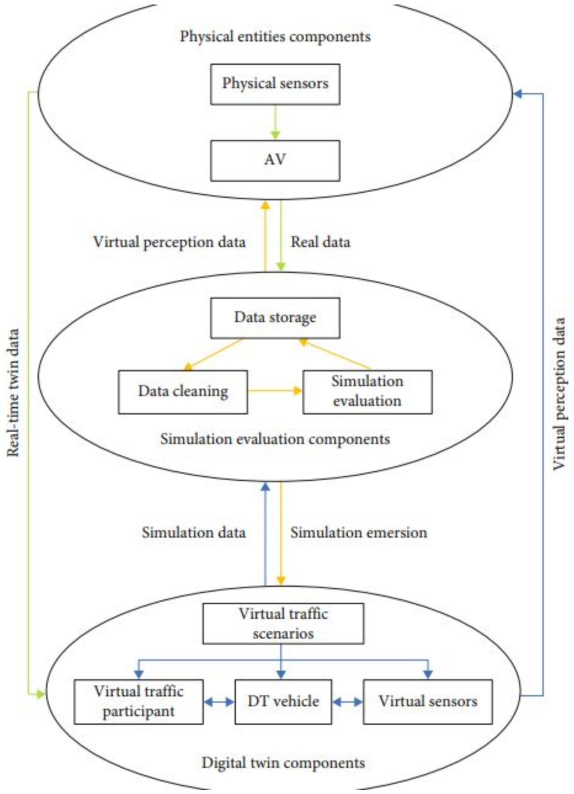
- Let’s design a digital twin for an autonomous vehicle
- We need physical components, data processing and evaluation components, and a digital replica of the autonomous vehicle
- The digital twin component generates virtual perception data to send back to the physical entity components
- The AV implements decision making and control based on the received data

Use Case: Autonomous Driving

SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

- We use real sensor devices to capture physical data and use it to implement the digitalization process from physical to digital space
- We let a real device (equipped with a controller) running over a real-testbed and collect measurements such as position, attitude, fault,..
- Data is transmitted in real-time to the DT via network communication and to the data processing and evaluation components for data storage
- The controller receives the virtual perception information generated by the DT and make decision and control via real test filed and real vehicle actuators to gain insights



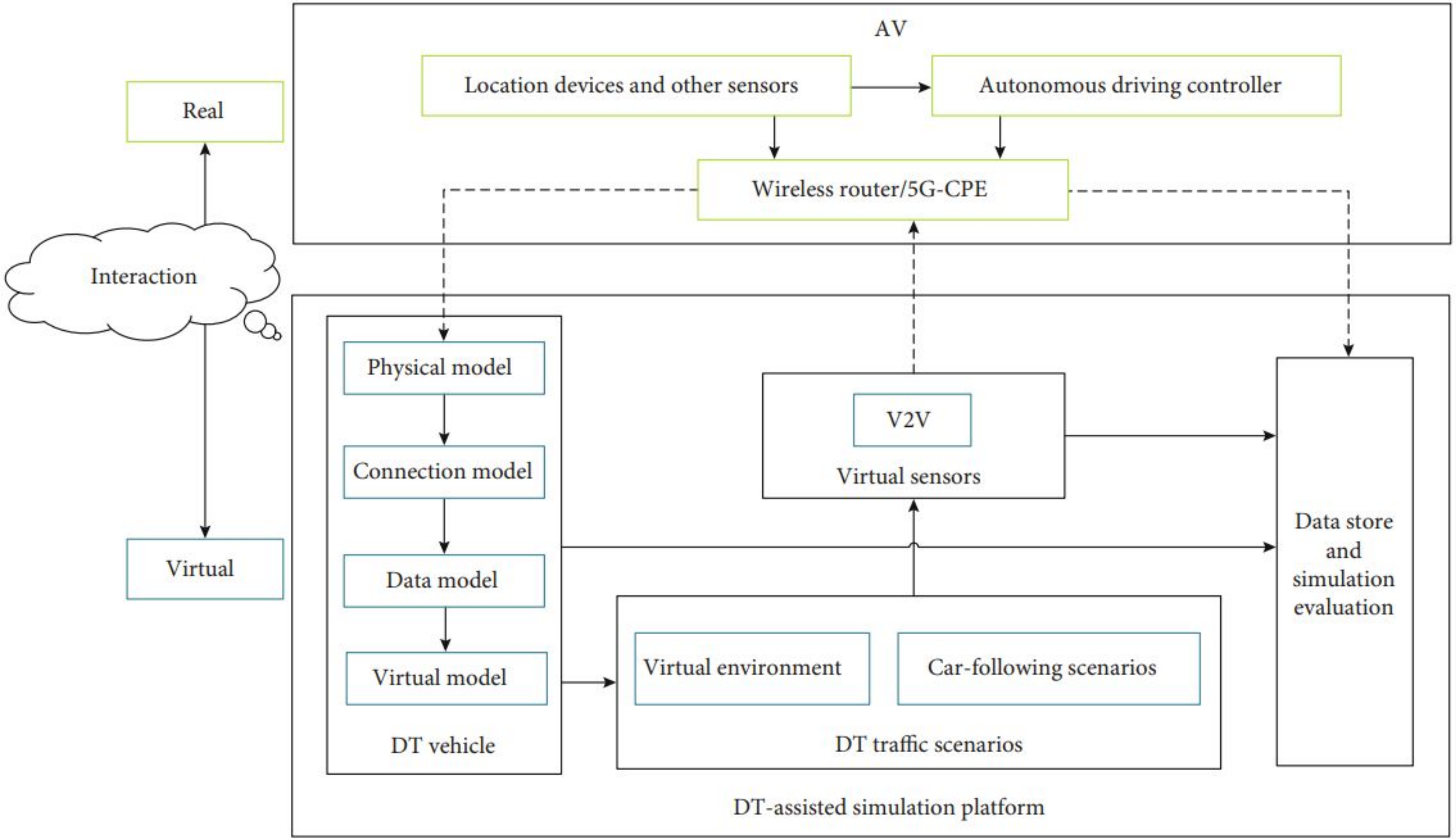
DT Components

- The components are digital descriptions of certain historical moments of current real-time state in physical entity components
- The DT and AV exchange real time data and need to maintain updated information to have the DT in line with what is actually happening to the AV
- The DT components can provide repeatable and extreme simulation test scenarios, such that the virtual sensors can sense such extreme information
- Th DT is also used as a visualization module

DT Components

SPRITZ Security & Privacy Research Group

Università degli Studi di Padova



DT Scenario

- The DT is divided into two parts: the virtual environment and the car-following scenarios
- The virtual environment needs to reflect the characteristics of a real environment from various aspects (very high accuracy)
- We use software for modeling of 3D environment (e.g., of a city)
- Very high accuracy means: gathering information from satellites on elevation data, vectorize data and model the city based on them, high definition render pipeline, import traffic scenarios

AV Replica

- The AV replica in the DT can divided into four levels: physical model, connection model, data model, and virtual model
- The physical model of the DT vehicle is defined to describe the motion characteristics of the AV to be tested
- The connection model of the DT vehicle realizes the synchronization between the AV to be tested and the DT vehicle
- The data model drives the virtual model to keep in sync with the behavior of the physical AV to be tested
- Virtual model is the visual model of the AV to be tested

Simulation Evaluation Components

- This component receives real-time data from the DT and the physical entities for storage and mining
- It can perform simulation evaluation based on the real-time data fed back
- The simulation results are stored in this components which can realize the rapid reproduction of dangerous accident scenarios and key simulation test processes

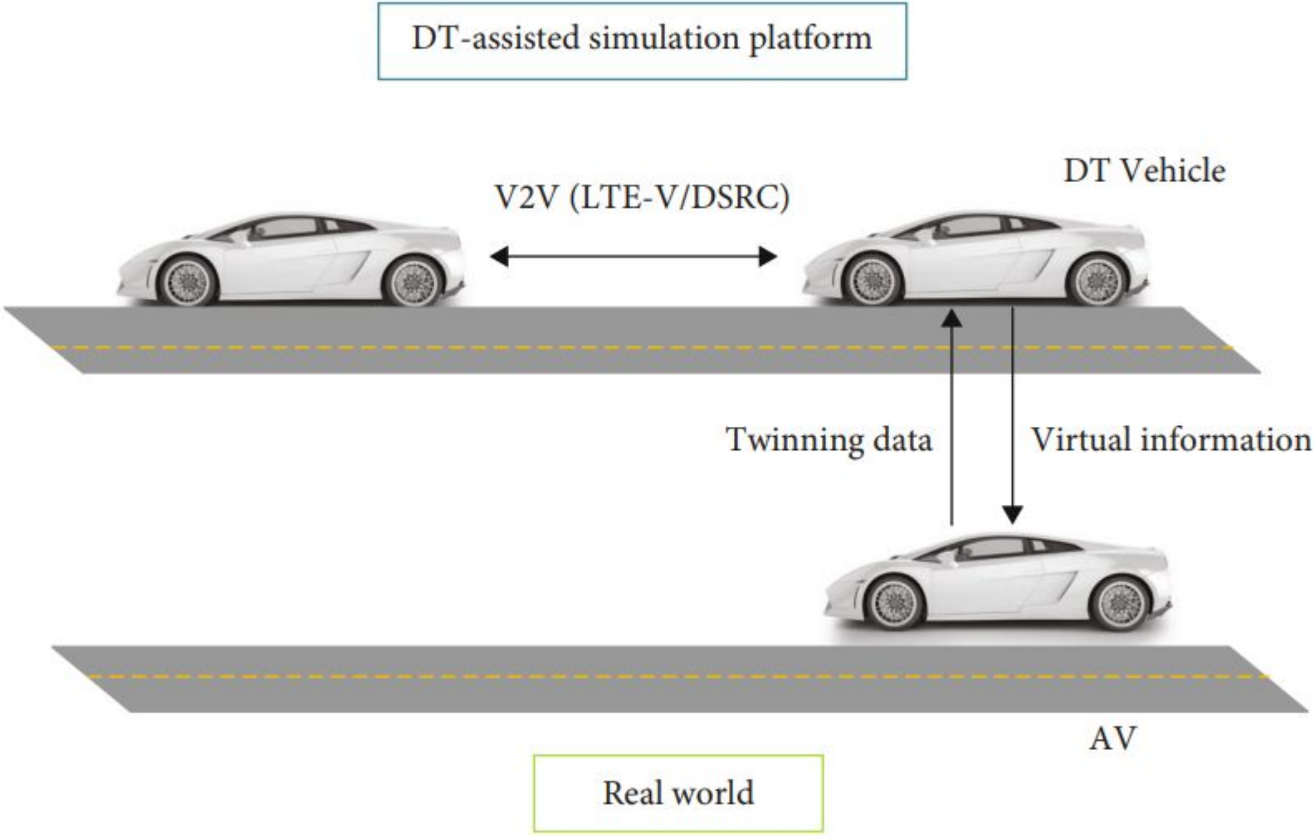
Car-Following Scenario

SPRITZ
Security & Privacy
Research Group

Università
degli Studi
di Padova

- We can use a DT to test breaking strategies of AVs without crashing cars
- In this case, we consider a platooning scenario, where a vehicle follows a platoon leader that suddenly breaks
- Time To Collision (TTC) represents the time it takes for the following vehicle to collide with the leading vehicle from the current state of motion
- We use this to verify the effectiveness of the model and policy

Car-Following Scenario



Car-Following Scenario

- The rear vehicle is represented by the DT which is a replica of the real AV and maintain a real-time information exchange
- We assume they use V2V communication, implemented via the DT communication module
- The DT vehicle sends the real-time virtual perception information obtained from the Preceding Vehicle (PV) to the AV in the physical world
- PV have four random motion states, that is stationary, accelerated motion, uniform motion, and decelerated motion

Car-Following Scenario

- The DT vehicle receives the speed, acceleration, and position information of PV through V2V
- Calculates the distance information between the two vehicles according to the position of the DT vehicle
- Acceleration and speed information of the two vehicles, as well as the distance between the two vehicles and the safe distance, and the TTC are calculated by collision avoidance strategies

Strategy Testing

- We can implement a collision avoidance strategy and test whether it works in different conditions thanks to the DT
- Implement the strategy and make the leader vehicle behave such that a collision would occur
- Run the AV based on the data obtained from the DT emulation
- Check whether there is a crash in the DT simulation

Consistency in Reasoning and Representation

- “what the physical asset projects must be equivalent to what its digital counterpart interprets and shows”
- We can use semantic description of languages to encode digital assets (ontology-based models) to avoid creating contradictory realities
- Interpret and derive conclusions at different granularity levels
- Related to consistency we can identify two sub requirements: fidelity and granularity

DTs for CPS Security

- While most of the DT available are used for maintenance, optimization, and simulation of CPS, DTs can also be used to increase security and safety
- A DT can run in parallel to a CPS to perform security and safety analysis while during operation
- This can be used to minimize the risk of security problems in the real domain, verifying whether a predefined action could bring to damages by looking at the digital domain
- State replication approach, Kalman filter

State Replication

- We consider a program P being a finite-state machine defined by a tuple $P := (X, x_0, U, Y, \delta, \lambda)$, where X is the finite set of states, $x_0 \in X$ denotes the initial state, $U := \{u_0, u_1, \ldots, u_{k-1}\}$ is the set of inputs, $Y := \{y_0, y_1, \ldots, y_{m-1}\}$ is the finite set of outputs, $\delta \colon X \times U \to X$ is the transition function, and $\lambda \colon X \times U \to Y$ is the output function
- The DT runs a program $\hat{P}$ that is functionally identical to the main one
- Thus, $\delta(x, u) = \hat{\delta}(\hat{x}, \hat{u}) \Leftrightarrow x' = \hat{x}'$ provided that $(x = \hat{x}) \wedge (u = \hat{u})$
- U^* and Y^* denote the set of all devices' possible inputs and outputs, respectively

State Replication

- We define the set S of stimuli, representing the roots of subsequent inputs corresponding to one DT (sensors values or HMI commands)

$$S := \{ z \in \hat{U} \mid z \in U \wedge z \notin Y^* \}$$

- $\hat{U}$ is the set of inputs of a DT, with $S \subset \hat{U}$ and $\hat{U}$ may contain elements of $\hat{Y}^*$ as well as inputs from users interacting with the DT
- We want to *replicate* stimuli, i.e., repeat the same stimulus s from the physical environment on a DT such that it leads to an identical program state x

State Replication

- We want to *replicate* stimuli, i.e., repeat the same stimulus s from the physical environment on a DT such that it leads to an identical program state x
- If any member of S would not satisfy the right operand of the logical conjunction $z \in U \wedge z \notin Y^*$, then the DT may receive the same input twice
- if such a stimulus is erroneously replicated, then the DT receives the stimulus in addition to an input that was a prior output

Active Monitoring

SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

- Continuously polling for state changes represent an active monitoring approach that might be used to capture x
- However this comes with drawbacks
- Monitoring all devices requires huge communication overhead that might impact on real-time communications
- The impact is also on devices, not only networks
- An attacker might tamper the response of a poll request

Passive State Replication

SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

- We implement a passive approach, where we track down the trigger of a state
- Objective is to understand which u constitutes a stimulus so that it can be fed to the transition function of the DT
- We look for stimuli coming from sources that are already available in the system to avoid adding any type of overhead
- We use devices that are already part of the CPS

Passive State Replication

SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

- Devices that are located in the CPS receive an input u that might trigger a state transition such that $\delta(x, u) = x'$
- Consider an input u to PLC causing a state transition defined by the control logic
- Since u must be considered as a determining factor that drives the state transition, it can be directly fed to the PLC's DT provided that u=s
- However more complicated state transitions might lead to problems if, for instance, it is not possible to distinguish u from s

Specification of the CPS

SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

- To solve the aforementioned challenges, we use the specification of the CPS to identify stimuli
- Let us define the partial function f that maps the stimuli indices I to stimuli of all DTs S^*, i.e., $f: U^* \cup Y^* \nrightarrow S^*$
- Then I is defined as $I := \{ j \in U^* \cup Y^* \mid f(j) \in S^* \}$
- We then observe j and check whether it belongs to I
- Since j belongs to I if and only if f(j) is defined, the value f(j) is fed to the respective DT provided that j is indeed in the set I

Example

SPRITZ Security & Privacy Research Group
Università degli Studi di Padova

- Consider a packaging line managed by a HMI that communicates with a PLC to control a conveyor belt
- External influences can be inferred by examining the associated role of components
- The HMI receives inputs from users and, since this is a root of subsequent actions, these can be classified as stimuli for the DT of the HMI
- However, since no input detection mechanism has been implemented, it is not easy to observe the existence of a user input

Example

- A user input generates an output in the format of a request (e.g., Modbus) making x indirectly observable by passively monitoring the network traffic to capture outputs of the HMI
- We can create f based on the specified logical network
  - Extract information on how devices communicate (including details on the used protocol)
  - Identify source and destination of packets and which program variable they affect
  - If an output y of the HMI has been classified as an outcome of a stimulus, f is defined

Intrusion Detection

SPRITZ
Security & Privacy
Research Group

Università
degli Studi
di Padova

- Assuming that the specification of the CPS correctly describes the correct behavior and that each DT follows the state of its physical counterpart, a divergence between the two worlds should be detectable
- Comparing the inputs and outputs of the physical environment with those of the virtual one reveal either device faults or malicious activity
- Thus, the IDS processes elements of the sets $U^*, Y^*, \hat{U}^*, \hat{Y}^*$

Intrusion Detection

SPRITZ
Security & Privacy
Research Group

Università
degli Studi
di Padova

- The comparison between $p \in U^* \cup Y^*$ and $v \in \hat{U}^* \cup \hat{Y}^*$ relies on predefined features
- For instance, network traffic can be compared based on source and destination MAC addresses, source and destination IP addresses, Modbus TCP/IP ADUs
- If there is a mismatch in one of these features, the IDS raises an alarm
- The feature selection process has a huge impact on IDS's performance

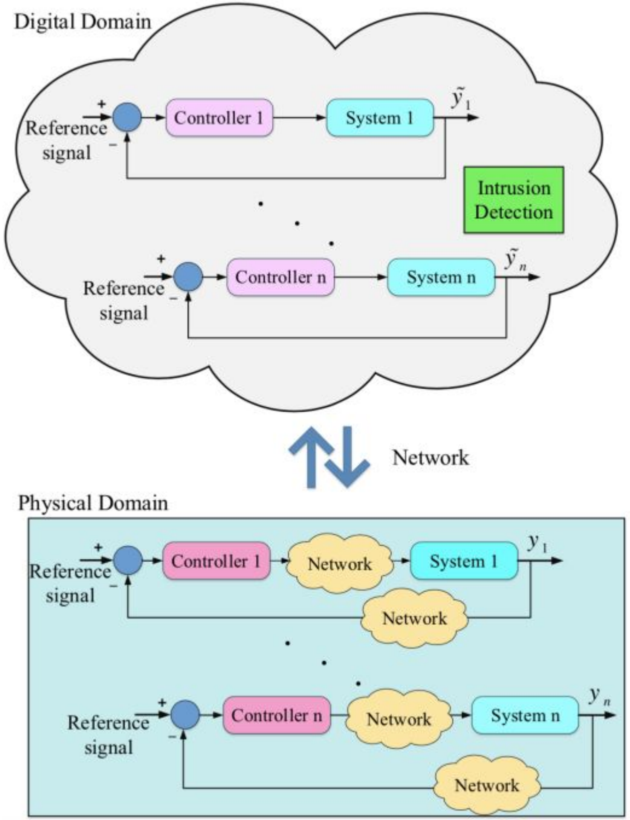
Kalman Filter-Based State Estimation

SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

- Let us consider an industrial control system, where different types of controllers regulate the production process
- We assume a DT for this system is deployed in the cloud, and there exists a connection between the two
- Such a connection creates possibilities for attacks besides those that already exist in the control system
- We want to develop an IDS able to detect both types of attacks

Kalman Filter-Based State Estimation



Scaling and Ramp Attacks

- Scaling attack: the measurement signal is manipulated via a scaling parameter during an attack period

$$y^*(t) = \begin{cases} y(t) & \text{for } t \notin \tau_a \\ (1+\lambda_s)\cdot y(t) & \text{for } t \in \tau_a \end{cases}$$

- Ramp attack: add a ramp signal to the actual measurement signal

$$y^*(t) = \begin{cases} y(t) & \text{for } t \notin \tau_a \\ y(t) + \lambda_r . t & \text{for } t \in \tau_a \end{cases}$$

Kalman Filter

- A Kalman filter is an algorithm that uses a series of noisy time measurements to produce an estimate of unknown variables that tend to be more accurate than those based on a single measurement
- For each time frame, it estimates the joint probability distribution
- Two phases:
  - Prediction phase: the algorithm produces estimates of the current state variables
  - Update phase: Once the next observation is available, update estimates via weighted average with more weight being given to estimates with greater certainty

Kalman Filter

- We use a Kalman filter to estimate the correct signals in the system by using inputs and outputs of the system
- We treat attacks as noisy components, which can be removed by the Kalman filter and therefore indicate change of behaviors
- We assume that there exists a simulated model of the real system that has been obtained easily by system identification algorithms
- Necessary to create an observable realization of this model to generate an observable state-space model

System Model

- We model our system as

$$x_{k+1} = Ax_k + Bu_k + Gw_k \qquad w \to N(0, Q)$$
$$y_k = Cx_k + Fv_k \qquad v \to N(0, R)$$

- Where x is the state vector, y is the output signal measured by sensors, u is the input signal generated by the controller, w is the process noise, and v is the measurement noise

Kalman Filter Design

- Time update part

$$1)\quad \hat{x}_{\mathrm{k|k-1}} = A_{\mathrm{k}} \hat{x}_{\mathrm{k-1k-1}} + B_{\mathrm{k}} u_{\mathrm{k}}$$

$$2)\quad P_{\mathrm{k|k-1}} = G_{\mathrm{k-1}} Q_{\mathrm{k-1}} G_{\mathrm{k-1}}^{\mathrm{T}} + A_{\mathrm{k-1}} P_{\mathrm{k-1|k-1}} A_{\mathrm{k-1}}^{\mathrm{T}}$$

- Measurement update

$$3)\quad K_{\mathrm{k}} = P_{\mathrm{k|k-1}} C_{\mathrm{k}}^{\mathrm{T}} \left(C_{\mathrm{k}} P_{\mathrm{k|k-1}} C_{\mathrm{k}}^{\mathrm{T}} + F_{\mathrm{k}} R_{\mathrm{k}} F_{\mathrm{k}}^{\mathrm{T}} \right)^{-1}$$

$$4)\quad \hat{x}_{\mathrm{k|k}} = \hat{x}_{\mathrm{k|k-1}} + K_{\mathrm{k}} \left(y_{\mathrm{k}} - C_{\mathrm{k}} \hat{x}_{\mathrm{k|k-1}} \right)$$

$$5)\quad P_{\mathrm{k|k}} = \left(I - \mathrm{K}_{\mathrm{k}} C_{\mathrm{k}} \right) P_{\mathrm{k|k-1}}$$

Kalman Filter Design

- Time update part

$$1)\quad \hat{x}_{\mathrm{k|k-1}} = A_{\mathrm{k}}\hat{x}_{\mathrm{k-1k-1}} + B_{\mathrm{k}}u_{\mathrm{k}}$$

Predicted state

Predicted covariance

$$2)\quad P_{\mathrm{k|k-1}} = G_{\mathrm{k-1}}Q_{\mathrm{k-1}}G_{\mathrm{k-1}}^{\mathrm{T}} + A_{\mathrm{k-1}}P_{\mathrm{k-1|k-1}}A_{\mathrm{k-1}}^{\mathrm{T}}$$

- Measurement update

Kalman Gain

$$3)\quad K_{\mathrm{k}} = P_{\mathrm{k|k-1}}C_{\mathrm{k}}^{\mathrm{T}}\left(C_{\mathrm{k}}P_{\mathrm{k|k-1}}C_{\mathrm{k}}^{\mathrm{T}} + F_{\mathrm{k}}R_{\mathrm{k}}F_{\mathrm{k}}^{\mathrm{T}}\right)^{-1}$$

Updated state estimate

$$4)\quad \hat{x}_{\mathrm{k|k}} = \hat{x}_{\mathrm{k|k-1}} + K_{\mathrm{k}}\left(y_{\mathrm{k}} - C_{\mathrm{k}}\hat{x}_{\mathrm{k|k-1}}\right)$$

Updated covariance estimate

$$5)\quad P_{\mathrm{k|k}} = \left(I - \mathrm{K_k}C_{\mathrm{k}}\right)P_{\mathrm{k|k-1}}$$

Kalman Filter Design

- Kalman filter requires the knowledge of the noise covariance matrices Q and R
- We can estimate them from the process and use methodologies such as particle swarm optimization to find the optimal estimation
- Using our designed filter, we can estimate the state variable x and consequently the system output as $\hat{y}_\mathrm{k} = C\hat{x}_\mathrm{k}$
- The residual signal is given by the difference between the stimate signal and the output signal $r_k = \tilde{y}_k - \hat{y}_k$

Detector

SPRITZ Security & Privacy Research Group

Università degli Studi di Padova

- We use a detector to to distinguish attack from noise and detect the occurrence of attacks
- Helps to make the effects of attacks and noises on the signal more prominent,filter impacts of noises, and prevent false alarms

$$h_k = \sup_{k-k_0<i<k} |r_i|, \quad \begin{cases} H_0: \text{ if } & h_k \leq \text{ threshold} \\ H_1: \text{ if } & h_k > \text{ threshold} \end{cases}$$

- Based on 68-95-99.7 rule, in a Gaussian distribution, 68.27%, 95.45%, and 99.73% of the values lie within one, two, and three standard deviations of the mean, respectively

Security Threats of Digital Twins

- DTs must be considered as critical systems
- We must care about CIA triad
- But we should also care about privacy issues of entities and localization of assets
- We consider two types of attack surfaces:
    - Physical, comprising attacks against access endpoints, CPS and IoT devices, communication infrastructure and facilities
    - Digital, comprising exploits against software, components for computing, and information systems