Supplementary material concerning the paper “Estimation and Prevention of Sensor Replacement Attacks in Supervisory Control Systems”

I. Proof of Theorem 1

Theorem 1: Given a plant G and a supervisor S, (1) S/G is strongly SR-estimable w.r.t. P_o^a, Σ_a, and X_u iff there exists a state b_o in $E_{S/G}$ such that $Fir(b_o) \subseteq X_u$; (2) S/G is weakly SR-estimable w.r.t. P_o, Σ_a, and X_u iff there exists a state b_o in $E_{S/G}$ such that $Fir(b_o) \cap X_u \neq \emptyset$, and for all states b_o' in $E_{S/G}$, $Fir(b_o') \cap (X \setminus X_u) \neq \emptyset$.

Proof: (1) ($\Leftarrow$) Suppose that there exists a state b_o in $E_{S/G}$ such that $Fir(b_o) \subseteq X_u$. For all states (x, q) in b_o, we have $x \in X_u$. According to the construction of $E_{S/G}$, for any decision string $\phi \in \Sigma_o \times (\Sigma_o \cup \{\varepsilon\})$ such that $f_e(b_{0,o}, \phi) = b_o$, we have that for all decision strings $\omega' \in P_o^{a-1}(\phi) \cap L(M_a)$, $f(x_0, \alpha(\omega')) \in X_u$. Then, there exists $\omega \in L(M_a)$ with $P_o^a(\omega) = P_o^a(\omega') = \phi$ such that the condition in Definition 1 holds, i.e., S/G is strongly SR-estimable w.r.t. P_o^a, Σ_a, and X_u.

($\Rightarrow$) Suppose that S/G is strongly SR-estimable w.r.t. P_o^a, Σ_a, and X_u. Then, there exists a decision string $\omega \in L(M_a)$ such that the condition in Definition 1 hold. Let $\phi = P_o^a(\omega)$. By Definition 3, it holds $\phi \in L(E_{S/G})$, i.e., there exists a state b_o such that $f_e(b_{0,o}, \phi) = b_o$. For any state (x, q) in b_o, there exists a decision string $\omega' \in P_o^{a-1}(\phi) \cap L(M_a)$ such that $f_a((x_0, q_0), \omega') = (x, q)$ and $x \in X_u$, i.e., $Fir(b_o) \subseteq X_u$.

(2) ($\Leftarrow$) Suppose that there exists a state b_o in $E_{S/G}$ such that $Fir(b_o) \cap X_u \neq \emptyset$. Then, there exists a state (x, q) in b_o such that $x \in X_u$. According to the construction of $E_{S/G}$, given a decision string $\phi \in \Sigma_o \times (\Sigma_o \cup \{\varepsilon\})$ such that $f_e(b_{0,o}, \phi) = b_o$, there exists a decision string $\omega \in P_o^{a-1}(\phi) \cap L(M_a)$ such that $f(x_0, \alpha(\omega)) \in X_u$, i.e., condition (1) in Definition 2 hold. Suppose that for all states b_o' in $E_{S/G}$, $Fir(b_o') \cap (X \setminus X_u) \neq \emptyset$, i.e., there exists a state (x', q') in b_o' such that $x' \notin X_u$. According to case (1), S/G is not strongly SR-estimable w.r.t. P_o^a, Σ_a, and X_u. By Definition 2, we conclude that S/G is weakly SR-estimable w.r.t. P_o^a, Σ_a, and X_u.

($\Rightarrow$) Suppose that S/G is weakly SR-estimable w.r.t. P_o^a, Σ_a, and X_u. Then, there exists a decision string $\omega \in L(M_a)$ such that $f(x_0, \alpha(\omega)) \in X_u$, and S/G is not strongly SR-estimable w.r.t. P_o^a, Σ_a, and X_u. Due to $P_o^a(\omega) \in L(E_{S/G})$, there exists a state (b_o, d_o) such that $f_e(b_{0,o}, P_o^a(\omega)) = b_o$. By $f(x_0, \alpha(\omega)) \in X_u$, there exists a state (x, q) in b_o such that $x \in X_u$ (i.e., $Fir(b_o) \cap X_u \neq \emptyset$). By case (1), for any state b_o' in $E_{S/G}$, there exists (x', q') in b_o' such that $x' \notin X_u$, i.e., $Fir(b_o') \cap (X \setminus X_u) \neq \emptyset$. This completes the proof. ■

II. Proof of Theorem 2

Theorem 2: Given $E_{S/G}$ w.r.t. S/G, (1) let $L_{sb} \neq \emptyset$ and $BS = BS_s$. An SSR-safe DI-function D exists if and only if the DIS Υ^{BS} w.r.t. $E_{S/G}$ and BS is not an empty automaton; (2) let $L_{sb} \cup L_{wb} \neq \emptyset$ and $BS = BS_s \cup BS_w$. An SR-safe DI-function D exists if and only if the DIS Υ^{BS} w.r.t. $E_{S/G}$ and BS is not an empty automaton.

Proof: (1) ($\Leftarrow$) If the DIS Υ^{BS} is not the empty automaton, there exists an SSR-safe DI-function D that can be synthesized from Υ^{BS} according to Proposition 1.

($\Rightarrow$) If an SSR-safe DI-function D exists, it holds that D can be synthesized from the DIS based on Proposition 1. Then, the DIS is not an empty automaton. Thus, this theorem holds.

(2) It can be proved in the same way as (1). ■

III. Construction of a DIS

We briefly review the construction of an “All insertion structure” in [21]. Let $\mathscr{D} = (M_1, \Sigma, \delta_1, m_{0,1})$ and $\mathscr{A} = (M_2, \Sigma, \delta_2, m_{0,2})$ be two automata.

In [21], the set of all information states is denoted by $\mathcal{Q} = M_1 \times M_2$, and the AIS is the tuple:

$$\text{AIS} = (Y, Z, \Sigma, M_1, f_{\text{AIS},yz}, f_{\text{AIS},zy}, y_0)$$

where Σ is the set of events in $\mathscr{A}$. M_1 is the set of states in $\mathscr{D}$. $Y \subseteq \mathcal{Q}$ is the set of Y-states. $Z \subseteq \mathcal{Q} \times \Sigma$ is the set of Z-states. Let $\mathcal{Q}(z)$, $\mathcal{E}(z)$ denote the information state component and event component of $z \in Z$ respectively, so that $z = (\mathcal{Q}(z), \mathcal{E}(z))$. $f_{\text{AIS},yz} : Y \times \Sigma \to Z$ is the transition function from Y-state to Z-state. For $y = (m_1, m_2) \in Y$, $\sigma \in \Sigma$, we have: $f_{\text{AIS},yz}(y, \sigma) = z \Rightarrow [\delta_2(m_2, \sigma)!] \wedge [\mathcal{Q}(z) = y] \wedge [\mathcal{E}(z) = \sigma]$. $f_{\text{AIS},zy} : Z \times M_1 \to Y$ is the transition function from Z-state to Y-state. For $z = ((m_1, m_2), \sigma) \in Z$, $m_1' \in M_1$, we have: $f_{\text{AIS},zy}(z, m_1') = y \Rightarrow [\exists s \in \Sigma^* \text{s.t.} \delta_1(m_1, s) = m_1'] \wedge [\delta_1(m_1', \sigma)!] \wedge [y = (\delta_1(m_1', \sigma), \delta_2(m_2, \sigma))]$. $y_0 \in Y$ is the unique initial Y-state, where $y_0 = (m_{0,1}, m_{0,2})$.

Given two automata $\mathscr{D} = (M_1, \Sigma, \delta_1, m_{0,1})$ and $\mathscr{A} = (M_2, \Sigma, \delta_2, m_{0,2})$, the construction procedure for the AIS consist of two steps: (1) obtaining the AIS$_{pre}$, and (2) obtaining the AIS. Based on $\mathscr{D}$ and $\mathscr{A}$, the game-like structure AIS$_{pre}$ can be obtained by Algorithm 1 in [21]. By Algorithm 2 in [21], the AIS can be obtained by pruning away all the inappropriate insertion choices in the AIS$_{pre}$.

```
Algorithm 1: Construction AIS$_{pre}$ in [21]
   Input: $\mathscr{D} = (M_1, \Sigma, \delta_1, m_{0,1})$ and $\mathscr{A} = (M_2, \Sigma, \delta_2, m_{0,2})$
   Output: $\mathrm{AIS}_{pre} = (Y, Z, \Sigma, M_1, f_{\mathrm{AIS}_{pre},yz}, f_{\mathrm{AIS}_{pre},zy}, y_0)$
 1 $y_0 := (m_{0,1}, m_{0,2})$, $Y := \{y_0\}$, $Z := \emptyset$;
 2 for all $y = (m_1, m_2) \in Y$ that have not been examined do
 3     for $\sigma \in \Sigma$ do
 4         if $\delta_2(m_2, \sigma)$ is defined then
 5             $f_{\mathrm{AIS}_{pre},yz}(y, \sigma) := (y, \sigma)$;
 6             $Z := Z \cup \{f_{\mathrm{AIS}_{pre},yz}(y, \sigma)\}$;
 7 for all $z = (y, \sigma) = ((m_1, m_2), \sigma) \in Z$ that have not been examined do
 8     for $m' \in M_1$ do
 9         if $\delta_1(m', \sigma)$ is defined and $\exists t \in \Sigma^*$ such that $m' = \delta_1(m_1, t)$ then
10             $f_{\mathrm{AIS}_{pre},zy}(z, m') := (\delta_1(m', \sigma), \delta_2(m_2, \sigma))$;
11             $Y := Y \cup \{f_{\mathrm{AIS}_{pre},zy}(z, m')\}$;
12 Go back to step 2; repeat until all accessible part has been built;
```

```
Algorithm 2: Construct AIS in [21]
   Input: $\mathrm{AIS}_{pre} = (Y, Z, \Sigma, M_1, f_{\mathrm{AIS}_{pre},yz}, f_{\mathrm{AIS}_{pre},zy}, y_0)$
   Output: $\mathrm{AIS} = (Y, Z, \Sigma, M_1, f_{\mathrm{AIS},yz}, f_{\mathrm{AIS},zy}, y_0)$
 1 Obtain an automaton as
     $A = (Y \cup Z, \Sigma \cup M_1, f_{\mathrm{AIS}_{pre},yz} \cup f_{\mathrm{AIS}_{pre},zy}, y_0)$;
 2 Mark all the $Y$-states in $A$;
 3 Let $\Sigma$ be uncontrollable and $M_1$ be controllable;
 4 Trim $A$ and let $A_{trim}$ be the specification automaton;
 5 Obtain the AIS as the automaton obtained from
     $[L_m(A_{trim})]^{\uparrow C}$ w.r.t. $L(A)$ by following the standard $\uparrow C$ algorithm in [22];
 6 return the AIS as
     $\mathrm{AIS} = (Y, Z, \Sigma, M_1, f_{\mathrm{AIS},yz}, f_{\mathrm{AIS},zy}, y_0)$;
```

Next, we integrate these two algorithms and transform them into one algorithm (Algorithm 3 in the supplementary material) to build a DIS in our work. Given an attacker estimator $E_{S/G}$ and a bad state set $BS \in \{BS_s, BS_s \cup BS_w\}$, we first obtain a safe estimator $E^{BS}_{S/G}$ w.r.t. S/G and BS by removing all the states in BS from $E_{S/G}$ and keeping the accessible part in step 1. Step 2 initializes the sets I_y and I_z. Steps 3–7 and 8–12 define the transitions from Y-states to Z-states and the transitions from Z-states to Y-states, respectively. In step 13, an automaton $\Upsilon = (I_y \cup I_z, \Xi_o \cup B^{BS}_o, f_{pre,yz}, f_{pre,zy}, y_0)$ is built. We prune away all inadmissible insertion cases that lead to deadlock at Z-states in Υ by steps 14–17. In step 18, a DIS is constructed. Given an estimator with $|B_o|$ states and the set of observable decision events $\Xi_o = \Sigma_o \times (\Sigma_o \cup \{\varepsilon\})$, the obtained DIS has at most $(|\Xi_o| + 1)|B_o|^2$ states, and the computational complexity for constructing the DIS is $\mathcal{O}(|B_o|^6)$ by referring to [20], [21].

```
Algorithm 3: Construction of DIS
   Input: An attacker estimator $E_{S/G} = (B_o, \Xi_o, f_e, b_{0,o})$
          and a bad state set $BS \in \{BS_s, BS_s \cup BS_w\}$
   Output: A DIS $\Upsilon^{BS} = (I_y, I_z, \Xi_o, B^{BS}_o, f_{yz}, f_{zy}, y_0)$
 1 Construct a safe estimator
     $E^{BS}_{S/G} = (B^{BS}_o, \Xi_o, f^{BS}_e, b_{0,o})$ by removing all the
     sets in $BS$ from $E_{S/G}$ and keeping the accessible part;
 2 $I_y := \{y_0\} = \{(b_{0,o}, b_{0,o})\}$, $I_z := \emptyset$;
 3 for all $i_y = (b_{o1}, b_{o2}) \in I_y$ that have not been examined do
 4     for $\sigma_{\sigma'} \in \Xi_o$ do
 5         if $f_e(b_{o2}, \sigma_{\sigma'})!$ then
 6             $f_{pre,yz}(i_y, \sigma_{\sigma'}) := (i_y, \sigma_{\sigma'})$;
 7             $I_z := I_z \cup \{f_{pre,yz}(i_y, \sigma_{\sigma'})\}$;
 8 for all $i_z = (i_y, \sigma_{\sigma'}) = ((b_{o1}, b_{o2}), \sigma_{\sigma'}) \in I_z$ that have not been examined do
 9     for $b'_{o1} \in B^{BS}_o$ do
10         if $f^{BS}_e(b'_{o1}, \sigma_{\sigma'})!$ and there exists $\omega \in \Xi^*_o$ such that $b'_{o1} = f^{BS}_e(b_{o1}, \omega)$ then
11             $f_{pre,zy}(i_z, b'_{o1}) :=$
                 $(f^{BS}_e(b'_{o1}, \sigma_{\sigma'}), f_e(b_{o2}, \sigma_{\sigma'}))$;
12             $I_y := I_y \cup \{f_{pre,zy}(i_z, b'_{o1})\}$;
13 Go back to step 2, repeat until all accessible part has been built, and build an automaton as
     $\Upsilon = (I_y \cup I_z, \Xi_o \cup B^{BS}_o, f_{pre,yz}, f_{pre,zy}, y_0)$;
14 Mark all the $Y$-states in $\Upsilon$;
15 Let $\Xi_o$ be uncontrollable and $B^{BS}_o$ be controllable;
16 Trim $\Upsilon$ and let $\Upsilon_{trim}$ be the specification automaton;
17 Construct DIS $\Upsilon^{BS}$ as the automaton obtained from
     $[L_m(\Upsilon_{trim})]^{\uparrow C}$ w.r.t. $L(\Upsilon)$ by using the standard
     $\uparrow C$ algorithm in [22];
18 return DIS as $\Upsilon^{BS} = (I_y, I_z, \Xi_o, B^{BS}_o, f_{yz}, f_{zy}, y_0)$;
```

Algorithm 3 is an integrated version of Algorithms 1 and 2 in [21]. Intuitively, we first take the automata $E^{BS}_{S/G}$ and $E_{S/G}$ as the input of Algorithm 1, i.e., substituting the automata $E^{BS}_{S/G}$ and $E_{S/G}$ for the automata $\mathscr{D}$ and $\mathscr{A}$, respectively. Then, we go directly to the step 1 of Algorithm 2 to obtain an automaton $\Upsilon = (I_y \cup I_z, \Xi_o \cup B^{BS}_o, f_{pre,yz}, f_{pre,zy}, y_0)$. Finally, we build a DIS Υ^{BS} by pruning away all inadmissible insertion cases that lead to deadlock at Z-states in Υ.

IV. Figure of Example 5

We build an automaton Υ based on $E_{S/G}$ and $BS = BS_s$ as shown in Fig. 1 of this supplementary material. All dashed states and arcs should be pruned since they correspond to inadmissible insertion cases, and a DIS Υ^{BS} is obtained. We use $\sigma_{\sigma'}$ and $\sigma_{1\sigma'_1}$ to represent any event in decision event sets $\{b_b, b_\varepsilon, b_d\}$ and $\{b_b, b_\varepsilon, b_d, d_d, d_\varepsilon, d_b\}$, respectively. For instance, we use a transition $f_{yz}(\chi_6\chi_6, \sigma_{\sigma'}) = (\chi_6\chi_6, \sigma_{\sigma'})$ to briefly represent the transitions $f_{yz}(\chi_6\chi_6, b_b) = (\chi_6\chi_6, b_b)$, $f_{yz}(\chi_6\chi_6, b_\varepsilon) = (\chi_6\chi_6, b_\varepsilon)$, and $f_{yz}(\chi_6\chi_6, b_d) = (\chi_6\chi_6, b_d)$.

V. Figure of Example 6

In Fig. 2 of this supplementary material, an automaton Υ is constructed based on $E_{S/G}$ and $BS = BS_s \cup BS_w$, and a DIS Υ^{BS} is obtained by removing all the dashed states and arcs in Υ.

VI. Setting of Prevention Module

This section demonstrates the specific setting of the proposed prevention module, which includes the following three parts.

A. Communication Topology Modeling

Fig. 3 illustrates the precise location of the prevention module within the industrial control system data flow, which is located within the Trusted Zone and operates before encryption and signing. Simultaneously, it demonstrates how digital twin technology is used to support the prevention module in generating high-fidelity fake data. This topology model is divided into three main zones: the Physical & Trusted Zone, the Network Transmission Zone, and the Attacker View Zone.

(1) Physical & Trusted Zone

This is the core secure area of the system, containing a real closed-loop system, a digital twin server, and a honeypot.

Digital twin server: By receiving feedback regarding the real system's structure, behavior, and features, as well as the information of the fake system's states, it ensures that the output of the fake system is physically sound and temporally consistent with the original system (high fidelity).

Prevention module (with obfuscation mechanism): It acquires data generated by the fake system and, based on a pre-synthesized offline insertion automaton, determines whether to transmit real data or insert fake events.

Security processor - MAC/Encrypt: Located after the prevention module. It receives the modified data sequence and uses a legitimate key stored within the trusted zone to generate a valid MAC and perform encryption and packaging.

(2) Network Transmission Zone

Encrypted and signed data packets are transmitted through the network. The data packets transmitted in this zone are completely legitimate in form.

(3) Attacker View Zone

An attacker conducts reconnaissance through the network and intercepts data packets. Since the data packets carry valid MACs, they pass standard protocol integrity checks. After decryption, the attacker's observations have been tampered with by the prevention module.

Based on the modified observations, the attacker derives an incorrect result of attack estimation (mistakenly believing the system has no attack value or its state is ambiguous), thereby abandoning the attack or delaying action.

B. Honeypot Fidelity Analysis

We construct a fake system with the same automaton structure as the real system and utilize digital twin technology to mirror the state and behavior of the real system in real time. By dynamically adjusting features such as device parameters and response latency, we strive to achieve behavioral indistinguishability from the attacker's perspective during the reconnaissance phase. We also acknowledge that an attacker might identify inconsistencies using redundant information or side channels (e.g., power consumption, timing). However, our current work primarily targets the reconnaissance phase scenario, assuming the attacker can only access data from partially compromised sensors and lacks multi-source verification capabilities. The aim is to delay the attacker's attack decisions by increasing the uncertainty of the attack result prediction.

C. Evaluation under Integrity-Protected Environments

The core of our strategy lies in deploying the prevention module with an obfuscation mechanism within the system's trusted zone (e.g., a location before data encryption). Acting as an authorized defense component, this module possesses legitimate keys, allowing it to re-calculate and append valid MACs after modifying the data, enabling the tampering to bypass the attacker's protocol integrity checks. Meanwhile, to ensure temporal consistency, the prevention module generates timestamps maintaining strict temporal monotonicity for each fake event, avoiding detection as anomalies or replay attacks. Furthermore, we ensure that all strings (event sequences) generated by the obfuscation mechanism belong to the existing safe behaviors of the system model, thereby enabling them to pass the system's integrity verification.

D. Deployment Feasibility and Synchronization Costs

The deployment of the prevention module requires certain prerequisites, such as sensor/actuator devices supporting secure communication protocols. It also assumes that an attacker cannot physically damage the hardware or directly access the raw sensor data of the real system. On this basis, we clarify that the construction of the prevention module (i.e., the synthesis of the insertion mechanism) is performed offline, that is, a process that can be computationally intensive without impacting real-time performance. Conversely, the execution (i.e., inserting fake events) is performed online. Based on the synthesized insertion automaton, the online execution is reduced to simple state traversal operations and state transitions. Its computational complexity is linear with respect to the number of system states and events. Therefore, the computational load fits well within the standard scanning cycles of modern controllers, making deployment feasible even on resource-constrained devices.

The obfuscation (insertion) mechanism in our proposed prevention module essentially introduces an interface in the communication link to modify system output, which may theoretically lead to issues such as communication latency and state inconsistency. Since our current work is based on the discrete event systems framework modeled by automata, we

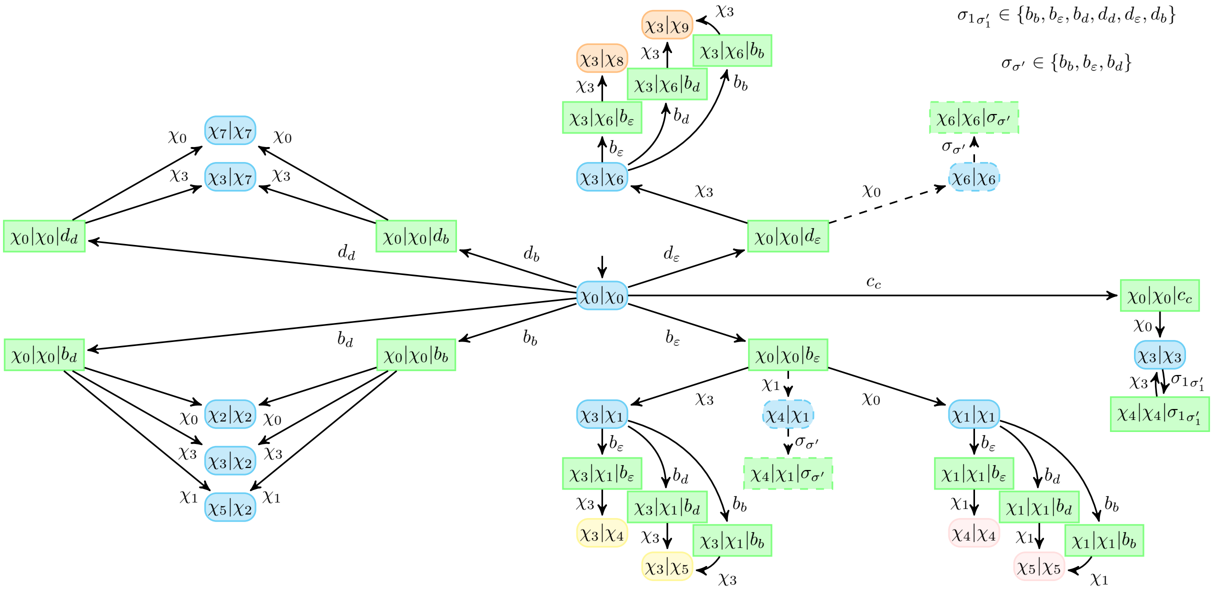
$\sigma_{1\sigma'_1} \in \{b_b, b_\varepsilon, b_d, d_d, d_\varepsilon, d_b\}$
$\sigma_{\sigma'} \in \{b_b, b_\varepsilon, b_d\}$
$\chi_0|\chi_0$
$\chi_0|\chi_0|d_d$
$\chi_0|\chi_0|d_b$
$\chi_0|\chi_0|d_\varepsilon$
$\chi_0|\chi_0|b_d$
$\chi_0|\chi_0|b_b$
$\chi_0|\chi_0|b_\varepsilon$
$\chi_0|\chi_0|c_c$
$\chi_7|\chi_7$
$\chi_3|\chi_7$
$\chi_2|\chi_2$
$\chi_3|\chi_2$
$\chi_5|\chi_2$
$\chi_3|\chi_6$
$\chi_3|\chi_6|b_\varepsilon$
$\chi_3|\chi_6|b_d$
$\chi_3|\chi_6|b_b$
$\chi_3|\chi_8$
$\chi_3|\chi_9$
$\chi_6|\chi_6$
$\chi_6|\chi_6|\sigma_{\sigma'}$
$\chi_3|\chi_3$
$\chi_4|\chi_4|\sigma_{1\sigma'_1}$
$\chi_3|\chi_1$
$\chi_3|\chi_1|b_\varepsilon$
$\chi_3|\chi_1|b_d$
$\chi_3|\chi_1|b_b$
$\chi_3|\chi_4$
$\chi_3|\chi_5$
$\chi_4|\chi_1$
$\chi_4|\chi_1|\sigma_{\sigma'}$
$\chi_1|\chi_1$
$\chi_1|\chi_1|b_\varepsilon$
$\chi_1|\chi_1|b_d$
$\chi_1|\chi_1|b_b$
$\chi_4|\chi_4$
$\chi_5|\chi_5$
d_d d_b d_ε b_d b_b b_ε c_c
χ_0 χ_1 χ_3 $\sigma_{\sigma'}$ $\sigma_{1\sigma'_1}$

Fig. 1. A DIS w.r.t. $\boldsymbol{E_{S/G}}$ and $\boldsymbol{BS_s}$ in Example 5.

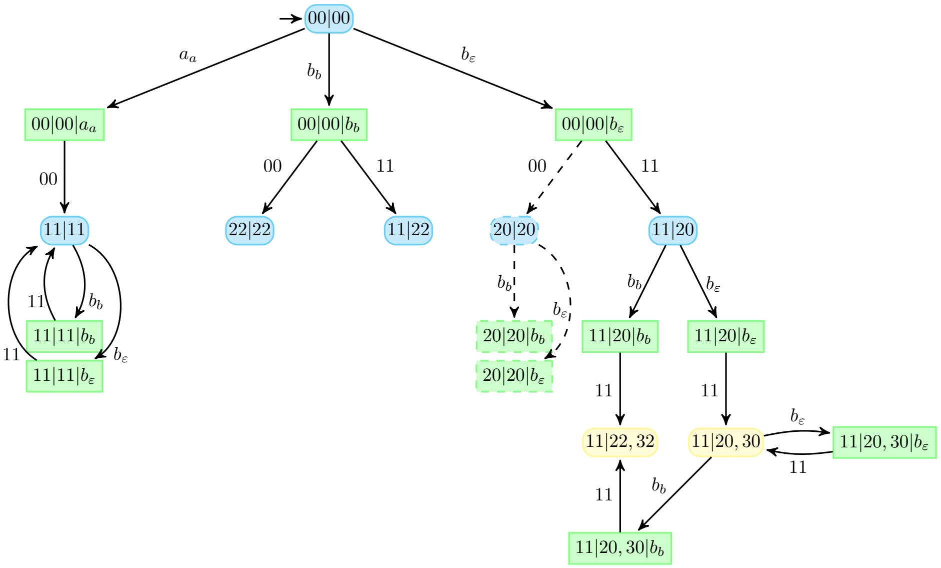

Fig. 2. A DIS w.r.t. $\boldsymbol{E_{S/G}}$ and $\boldsymbol{BS_s} \cup \boldsymbol{BS_w}$ in Example 6.

have not explicitly focused on quantitative time delay issues in this paper. Of course, we acknowledge that this is a critical consideration in networked control systems. Our future work will extend to networked discrete event systems, which will require calculating the execution overhead of the insertion mechanism and considering communication delays. To address this, we plan to define the “cost” as the maximum time delay introduced by the obfuscation operation. By trading off system security against execution overhead, we aim to ensure the synchronization cost remains within an acceptable range.

E. Robustness Analysis of the deception under adversarial adaptation

We have recognized that attackers may vary in sophistication. For example, an attacker unaware of a prevention module will likely follow a fixed attack strategy, whereas an intelligent attacker might indeed attempt to adjust the attack strategy upon suspecting the presence of the prevention module. We clarify that as long as an insertion function satisfying the property of SR-safety can be synthesized, such a function can guarantee that the attacker cannot uniquely determine the system state even under adaptive attack strategies. By ensuring the possible state estimates of the attacker include only safe states, the insertion function maintains the effectiveness of the prevention module.

References

[1] P. J. Ramadge and W. M. Wonham, “Supervisory control of a class of discrete event processes,” *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, Jan. 1987.

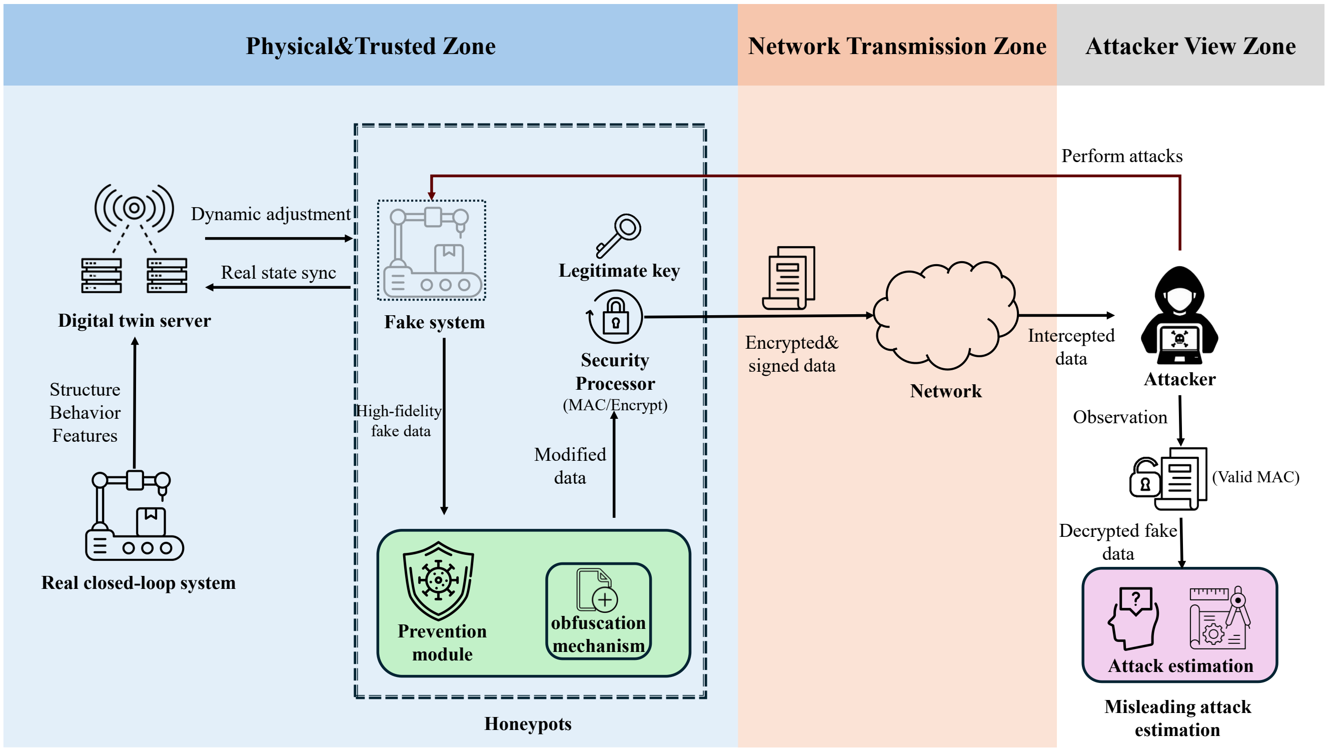

Fig. 3. Communication topology modeling.

[2] C. N. Hadjicostis, S. Lafortune, F. Lin, and R. Su, “Cybersecurity and supervisory control: A tutorial on robust state estimation, attack synthesis, and resilient control,” in *Proc. 61st IEEE Conf. Decis. Control (CDC)*, Cancun, Mexico, Dec. 2022, pp. 3020–3040.

[3] Q. Zhang, C. Seatzu, Z. Li, and A. Giua, “Selection of a stealthy and harmful attack function in discrete event systems,” *Sci. Rep.*, vol. 12, no. 1, p. 16302, Sep. 2022.

[4] Y. Li, C. N. Hadjicostis, N. Wu, and Z. Li, “Error- and tamper-tolerant state estimation for discrete event systems under cost constraints,” *IEEE Trans. Autom. Control*, pp. 1–8, Feb. 2023, doi: 10.1109/TAC.2023.3239590.

[5] R. Meira-Góes, E. Kang, R. H. Kwong, and S. Lafortune, “Stealthy deception attacks for cyber-physical systems,” in *Proc. 56th IEEE Conf. Decis. Control (CDC)*, Melbourne, Australia, Dec. 2017, pp. 4224–4230.

[6] ——, “Synthesis of sensor deception attacks at the supervisory layer of cyber–physical systems,” *Automatica*, vol. 121, p. 109172, Nov. 2020.

[7] M. Wakaiki, P. Tabuada, and J. P. Hespanha, “Supervisory control of discrete-event systems under attacks,” *Dyn. Games Appl.*, vol. 9, no. 4, pp. 965–983, Dec. 2019.

[8] R. Meira-Góes, S. Lafortune, and H. Marchand, “Synthesis of supervisors robust against sensor deception attacks,” *IEEE Trans. Autom. Control*, vol. 66, no. 10, pp. 4990–4997, Jan. 2021.

[9] L. K. Carvalho, Y.-C. Wu, R. Kwong, and S. Lafortune, “Detection and mitigation of classes of attacks in supervisory control systems,” *Automatica*, vol. 97, pp. 121–133, Nov. 2018.

[10] R. Su, “Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations,” *Automatica*, vol. 94, pp. 35–44, Aug. 2018.

[11] R. Meira-Góes, H. Marchand, and S. Lafortune, “Dealing with sensor and actuator deception attacks in supervisory control,” *Automatica*, vol. 147, p. 110736, Nov. 2023.

[12] Y. Zhu, L. Lin, and R. Su, “Supervisor obfuscation against actuator enablement attack,” in *Proc. 18th European Control Conference (ECC)*, Napoli, Italy, Jun. 2019, pp. 1760–1765.

[13] S. Oliveira, A. B. Leal, M. Teixeira, and Y. K. Lopes, “Integrity of cyber-physical discrete event systems under covert actuator attacks,” *IFAC-PapersOnLine*, vol. 58, no. 1, pp. 198–203, 2024.

[14] R. Meira-Góes and S. Lafortune, “Moving target defense based on switched supervisory control: A new technique for mitigating sensor deception attacks,” *IFAC-PapersOnLine*, vol. 53, no. 4, pp. 317–323, 2020.

[15] T. Yadav and A. M. Rao, “Technical aspects of cyber kill chain,” in *Proc. Int. Symp. Secur. Comput. Commun.* Springer, 2015, pp. 438–452.

[16] S. Roy, N. Sharmin, J. C. Acosta, C. Kiekintveld, and A. Laszka, “Survey and taxonomy of adversarial reconnaissance techniques,” *ACM Comput. Surv.*, vol. 55, no. 6, pp. 1–38, Jun. 2023.

[17] L. Lin and R. Su, “Synthesis of covert actuator and sensor attackers,” *Automatica*, vol. 130, p. 109714, Aug. 2021.

[18] D. You, S. Wang, M. Zhou, and C. Seatzu, “Supervisory control of petri nets in the presence of replacement attacks,” *IEEE Trans. Autom. Control*, vol. 67, no. 3, pp. 1466–1473, Mar. 2021.

[19] Y.-C. Wu and S. Lafortune, “Synthesis of insertion functions for enforcement of opacity security properties,” *Automatica*, vol. 50, no. 5, pp. 1336–1348, May 2014.

[20] Y. Ji, Y.-C. Wu, and S. Lafortune, “Enforcement of opacity by public and private insertion functions,” *Automatica*, vol. 93, pp. 369–378, Jul. 2018.

[21] R. Liu and J. Lu, “Enforcement for infinite-step opacity and k-step opacity via insertion mechanism,” *Automatica*, vol. 140, p. 110212, Jun. 2022.

[22] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer Science & Business Media, 2009.

[23] J. Franco, A. Aris, B. Canberk, and A. S. Uluagac, “A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber-physical systems,” *IEEE Commun. Surveys Tuts.*, vol. 23, no. 4, pp. 2351–2383, Aug. 2021.

[24] G. K. Edwin, S. V. Ewards, G. J. W. Kathrine, G. M. Palmer, A. Bertia, and S. Vijay, “Honeypot based intrusion detection system for cyber physical system,” in *Proc. Int. Conf. Augmented Intell. Sustain. Syst. (ICAISS)*, Trichy, India, Nov. 2022, pp. 958–962.

[25] X. Li, C. N. Hadjicostis, and Z. Li, “Extended insertion functions for opacity enforcement in discrete-event systems,” *IEEE Trans. Autom. Control*, vol. 67, no. 10, pp. 5289–5303, Oct. 2022.

[26] X. Li, C. N. Hadjicostis, and Z. Li, “Opacity enforcement in discrete event systems using extended insertion functions under inserted language constraints,” *IEEE Trans. Autom. Control*, vol. 68, no. 11, pp. 6797–6803, Nov. 2023.

[27] R. Liu, J. Lu, Y. Liu, X. Yin, and C. N. Hadjicostis, “Opacity enforce-

ment via greedy privately-and-publicly known insertion functions," *IEEE Trans. Autom. Control*, vol. 69, no. 4, pp. 2500–2506, Apr. 2024.

[28] Z. He and Z. Li, "Supplementary material concerning the paper 'Estimation and Prevention of Sensor Replacement Attacks in Supervisory Control Systems'," 2024. [Online]. Available: https://github.com/ZhaoyangHe-MUST/Supplementary-material/blob/main/generic-color.pdf