Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Matrix-Based Modeling of Discrete Event Systems under Event Insertion and Replacement Attacks

Yongyi Yan

yyyan@mail.nankai.edu.cn

Henan University of Science and Technology

Pengfei Meng

Henan University of Science and Technology

Jumei Yue

Henan University of Science and Technology

Xiaona Song

Henan University of Science and Technology

Zengqiang Chen

Nankai University

Research Article

Keywords: Logical systems, logical networks, finite-valued systems, finite state machines, semi-tensor product, Boolean networks

Posted Date: March 5th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8861876/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Matrix-Based Modeling of Discrete Event Systems under Event Insertion and Replacement Attacks

Yongyi Yan[1], Pengfei Meng[1], Jumei Yue[2], Xiaona Song[1], Zengqiang Chen[3],

1. College of Information Engineering, Henan University of Science and Technology, Henan, 471023, China;

2. College of Agricultural Equipment Engineering, Henan University of Science and Technology, Henan, 471023, China

3. College of Artificial Intelligence, Nankai University, Tianjin, 300350, China

Abstract: Existing research on discrete event system (DES) modeling and cybersecurity often lacks effective means to express events and dynamically characterize external attacks, thereby limiting the flexibility of security analysis. To overcome these limitations, this paper proposes a matrix-based event expression method that employs the semi-tensor product (STP) to convert auxiliary labels in DESs into a matrix form, leading to a more efficient mathematical characterization of dynamic behaviors. First, the event transmission capability of finite state machines (FSMs) is enhanced by formalizing event insertion and replacement between states as external attacks, thus improving the model's resilience against malicious interference. Subsequently, an external observation mechanism is designed to detect attacks during event transmission and quantify them via cost labels, thereby incorporating economic constraints into attacker behavior modeling. The results indicate that the proposed method improves the dynamic expressiveness of FSMs and supports the analysis and verification of DESs in cybersecurity contexts.

Keywords: Logical systems, logical networks, finite-valued systems, finite state machines, semi-tensor product, Boolean networks

1. Introduction

Discrete event systems (DESs) are widely employed in domains such as industrial control, cybersecurity, and automated systems. The theoretical framework of finite state machines (FSMs) serves as one of the core methodologies for modeling DESs. The rapid advancement of cyber-physical systems and the internet of things has raised

Corresponding author: Yongyi Yan. E-mail address: yyyan@mail.nankai.edu.cn

increasing concerns about the security of DESs. Consequently, modeling and detecting external malicious attacks, such as event tampering and false data injection, have emerged as active research areas [1-3]. However, traditional FSM models exhibit inherent limitations in capturing dynamic behaviors, which hinders the accurate characterization of complex attack patterns and leads to inefficient security verification. Therefore, enhancing the dynamic expressiveness of FSMs and optimizing their application in cybersecurity represents a critical challenge that requires urgent attention [4].

In recent years, many scholars have made important contributions to DESs modeling and security analysis. In terms of FSMs extension, Cassandras [5] proposed an event-driven state transition optimization method to improve system response speed, but did not consider the impact of external attacks. In the field of network security modeling, Kumar et al. [4] proposed a threat analysis method based on attack trees, but its integration with DESs is poor and cannot be directly applied to dynamic systems.

Existing research has primarily focused on control optimization, with comparatively less attention devoted to the dynamic characterization of security threats. In particular, the modeling of complex attack patterns, such as event tampering and man-in-the-middle attacks, remains insufficient. Therefore, the main limitations of current research include: the limited expressiveness of traditional FSMs in describing diverse attack patterns; insufficient flexibility in dynamic behavior expression, leading to low security verification efficiency [6][7]; and the insufficient application of existing matrix-based methods in security analysis, lacking efficient attack detection mechanisms [8]. On the other hand, although existing research has made certain progress in DESs modeling and network security, there are still the following key issues: (i) Insufficient expression of attacks. Traditional FSM models are difficult to describe complex attacks such as event insertion and replacement, limiting the accuracy of security analysis; (ii) Insufficient dynamic optimization. Existing methods have

complex calculations when adjusting system behaviors in real time, making it difficult to adapt to rapidly changing threat environments; (iii) The combination of matrix-based modeling and security analysis is rare, resulting in limited applicability of theoretical methods in engineering practice.

In terms of matrix-based modeling, the semi-tensor product (STP) theory has been applied to logical systems in recent years, expanding the expression form of FSMs from a methodological perspective and providing new ideas for the mathematical expression of DESs [9][10]. The STP of matrices not only retains the properties of traditional matrix multiplication but also has its own unique advantages, such as commutativity and compatibility, making the STP of matrices successfully applied in many fields, including logical systems [11], game networks [12], fuzzy control [13], image theory [14], power systems [15], biological systems [16], and other engineering problems [17].

Motivated by the aforementioned challenges and the potential of new mathematical tools, this paper undertakes the following research contributions: broaden the event transmission form of FSMs, enhance the model's ability to characterize malicious interference by defining the insertion and replacement of events between states as external attack types; adopt the STP method to convert additional labels in discrete events into matrix form, optimize the dynamic behavior expression of FSMs, and improve computational efficiency; verify the effectiveness of the method in network security scenarios [18-20]. The STP method can improve the efficiency of attack detection and system verification, providing a new perspective for the security modeling of dynamic systems [21-23]. Secondly, within the framework of DESs, two types of external attacks are considered, namely event insertion and replacement. For example, replacing some similar characters (for instance, replace the number 0 with the letter o, the letter I with the number 1, the letter a with the symbol @, or adjacent letters on the keyboard such as m and n). From the attacker's perspective, the purpose is to change the instructions in sensors or actuators while remaining hidden;

from the defender's perspective, the purpose is to perform state estimation, property analysis, interference detection, and supervisory control. The main contributions of this paper can be summarized as follows: (i) This paper mainly focuses on the state property analysis and state estimation of events under attacks. In the paper, the STP of matrices is used to model the replacement and insertion operations of transmitted events between states. (ii) The paper considers error diagnosis and interference detection, and verifies the modeling method through specific examples. Compared with the traditional diagnosability method (by calculating the number of error events after they occur), this work captures the existence of such a bound over the entire system language under a maximum cost C.

The structure of this paper is as follows: section 2 introduces the necessary preliminary knowledge, including the basic theories of STP and FSMs, and formally describes the research problem. Section 3 elaborates on the main results of this paper, including the construction of vectorized FSMs, the derivation of its dynamic equations, and attack modeling. Section 4 demonstrates the application process of the proposed method in specific scenarios through numerical examples. Section 5 conducts an in-depth comparison with existing research and presents the innovation and advantages of this method. Section 6 further discusses the research and looks forward to future directions. Finally, the seventh section gives the conclusion.

2. Preliminary knowledge and problem formulation

This section first presents a concise introduction to the foundational tools—namely, the semi-tensor product (STP) of matrices—and the research object, finite state machines (FSMs). It then formally formulates the research problem.

2.1 Semi-tensor product (STP) and finite state machine (FSM)

Definition 1[24]: Let $A \in R^{m\times n}$ and $B \times R^{p\times q}$ be two matrices. The STP of A and B, denoted by $A\hat{a}\ B$, is defined as

$$A\hat{a}\ B := (A \otimes I_{h/n})(B \otimes I_{h/p}), \tag{1}$$

where h is the least common multiple of n and p, and $\otimes$ is the Kronecker product of matrices, I_K is the $k \times k$ identity matrix.

Remark 1: (1) is the general form of the matrix STP. In practical applications, the following special forms of the matrix STP are widely used.

1. Let X and Y be a row vector of dimension np and a column vector of dimension p, respectively. Partition X into p qual blocks $X_1,...X_p$. The STP of X and Y denoted by $X \hat{a} Y$, is defined as

$$\begin{cases} X \hat{a}\ Y = \sum_{i=1}^{p} X_i y_i \in R^n, \\ Y^T \hat{a}\ X^T = \sum_{i=1}^{p} y_i (X_i)^T \in R^n. \end{cases}$$

where $y_i \in R$ is the i-th component of Y.

2. For matrices $A \in R^{m \times n}$ and $B \times R^{p \times q}$. If $p = r_1 n$, i.e., n is a factor of p, or $n = r_2 p$, i.e., p is a factor of n, then the STP of A and B, denoted by $D = \{D_{ij}\} = A \hat{a}\ B$, is defined as

$$D_{ij} = A^i \hat{a}\ B_j, i = 1,2,...,m, j = 1,2,...,q,$$

where A^i is the i-th row of A and B_j is the j-th column of B.

Remark 2: The classical multiplication of matrices is a special case of STP. When $n = p$, the latter degenerates into the former. STP not only retains almost all important properties of classical matrix multiplication, such as associativity and distributivity over addition, but also overcomes the inherent defects of classical multiplication, such as incomplete compatibility. For more properties of STP, refer to [24]. In this paper, the multiplication of matrices is defaulted to STP, and the symbol â is omitted unless otherwise emphasized.

Definition 2[24]: The swap matrix $W_{[m,n]}$ is an $mn \times mn$ matrix defined as follows: its rows and columns are labeled by double indices (i, j), the columns are arranged according to the index $Id(i, j; m, n)$, the rows (I, J) are labeled by double indices $Id(J, I; n, m)$ arranged according to the index, and the value of the element at position $[(I, J), (i, j)]$ is

$$w_{((I,J),(i,j))}=\delta_{i,j}^{I,J}=\begin{cases}1, & I=i \text{ and } J=j,\\ 0, & \text{otherwise.}\end{cases}$$

Proposition 1[24][25] (Pseudo-commutativity): Let X and Y be column vectors of dimensions m and n respectively, then the following pseudo-commutativity holds.

$$\begin{aligned} W_{[m,n]}\hat{a}\ X\hat{a}\ Y&=Y\hat{a}\ X,\\ W_{[n,m]}\hat{a}\ Y\hat{a}\ X&=X\hat{a}\ Y,\end{aligned} \tag{2}$$

Definition 3[26] (Finite state machine): A FSM M is a synchronous system with a finite set of states $X=\{x_1,x_2,...,x_n\}$, where x_i represents a state of the FSM $(i=1,2,...,n)$, a finite input alphabet $E=\{e_1,e_2,...,e_m\}$, where e_j represents an input event of the FSM $(j=1,2,...,m)$. A state transition function $f(x_i,e_j)=x_k$, where $x_i,x_k\in X,e_j\in E$. Formally, a FSM is a 5-tuple $M=(X,E,f,x_0,X_F)$, where $x_0\subset X$ is the initial state, $X_F\subset X$ is the set of marked states, and the machine stops in a state of X_F after reading an input string.

If $f:X\times E\to X$ is a function, i.e., for each state and input, the FSM has a unique next state, then it is a deterministic finite state machine (DFSM); otherwise, if $f:X\times E\to 2^X$ is a relation, it is a nondeterministic finite state machine (NFSM). If f is a total function, then the FSM is completely specified; otherwise, it is incompletely specified.

Theorem 1[6]: Let $F=[F_1,F_2...,F_m]$ be the transition structure matrix (TSM) of the FSM $M=(X,E,f,x_0,X_F)$, where $X=\{x_1,x_2,...,x_n\}$, $E=\{e_1,e_2,...,e_m\}$. The dynamic equation for state x_i of M reading the input sequence $e=e_1e_2\ldots e_t\in E^*$ is

$$x_j=F^t\hat{a}\ \delta_n^i\hat{a}\ u(t). \tag{3}$$

where $u(t)=\hat{a}_{j=1}^{t}\delta_m^j=\delta_m^1\hat{a}\ \delta_m^2\hat{a}\ ...\hat{a}\ \delta_m^t$, δ_m^j is the vector form of e_j $(j=1,2,...,t)$; δ_n^i is the vector form of x_i $(i=1,2,...,n)$; $F=F\hat{a}\ W_{[m,n]}$, $F=[F_1,F_2...,F_m]$.

$$F_i(s,t)=\begin{cases}1, & \delta_n^s\in f(\delta_n^t,\delta_m^i),\\ 0, & \text{otherwise.}\end{cases}$$

F is called the TSM of the FSM M.

Definition 4[6]: The dynamics of a FSM M reading the input sequence $e = e_1 e_2 \ldots e_t \in E^*$ at state x is $f(x,e) = f(f(...f(f(x,e_1),e_2)...),e_t)$, where denotes the set of finite strings on $E = \{e_1, e_2, ..., e_m\}$, and call $x \xrightarrow{e_1} x' \xrightarrow{e_2} x'' \xrightarrow{e_3} ...$ a path given the input string e with respect to the initial state x.

Definition 5[27]: Let $M = (X, E, f, x_0, X_F)$, where $X = \{x_1, x_2, ..., x_n\}$, $E = \{e_1, e_2, ..., e_m\}$. The state package of M is the set of states. The t-step reachable state package from state x_i reading the input sequence $u(t)$ is denoted as $D(x_i, u(t))$. When M reading $u(t)$ in state x_i, each state can be reached by reading $u(t)$, i.e., $D(x_i, u(t)) = \{x_{i_1}, x_{i_2}, ..., x_{i_k} \mid f(x_i, u(t)) = x_{i_j}, j = 1, 2, ..., k\}$.

Remark 3: (i) Let the t-step reachable state package from state x_i, $u(t)$ be $D(x_i, u(t)) = \{x_{i_1}, x_{i_2}, ..., x_{i_k}\}$, then the following equation holds.

$$f(x_i, u(t)) = x_{i_1}, f(x_i, u(t)) = x_{i_2}, ..., f(x_i, u(t)) = x_{i_t}.$$

(ii) For a DFSM, $|D(x_i, u(t))| = 1$ for any state x_i and any sequence $u(t)$.

(iii) The vector form of the state package $D(x_i, u(t)) = \{x_{i_1}, x_{i_2}, ..., x_{i_k}\}$ is $\delta_n^{i_1, i_2, ..., i_k}$.

In recent years, the following state transition dynamic equation (4) for FSMs has been developed and established [7][28][29], which is widely used to study the dynamic changes of FSMs.

$$x(t+1) = T^t x_0 u(t). \tag{4}$$

where $x(t+1)$ denotes the state at time step $(t+1)$, T^t denotes the dynamic matrix of the FSM, x_0 denotes the initial state of the FSM, and $u(t)$ denotes the vector form of the string with input length t. The states $x(t+1)$, x_0 and the input string $u(t)$ are all vectors. Specifically, for a FSM $M = (X, E, f, x_0, X_F)$, where $X = \{x_1, x_2, ..., x_n\}$, $E = \{e_1, e_2, ..., e_m\}$, state x_i is equivalent to δ_n^i, e_j is equivalent to δ_m^j, δ_n^i and δ_m^j are called vector forms of state x_i and input event e_j. Where δ_p^q is the q column vector of the identity matrix I_p. For the input sequence $u(t) = e_1 e_2 ... e_t$, its vector form is $\hat{a}_{j=1}^{t} \delta_m^j$.

The following are the notations used in this paper.

External interference label R_l: indicates whether the state has interference, $R_l = \{nr, r\}$, r indicates the state is interfered, nr indicates no interference.

Error detection label W_l: detects whether there is an erroneous event in the string, $W_l = \{nw, w\}$, nw indicates no erroneous event detected, w indicates an erroneous event detected.

Replacement label T_l: indicates the label carried after replacing the original event with a new event.

Insertion label Z_l: indicates the insertion of a new event.

Cost label C_l: the cost the attacker needs to bear after attacking the original event, i.e., $C_l = \{0,1,2,...,C\}$, maximum cost is C.

δ_n^i: the i-th column vector of the identity matrix I_n.

$\delta_n[i_1, i_2, ..., i_k]$: the compact form of the matrix $[\delta_n^{i_1}, \delta_n^{i_2}, \cdots, \delta_n^{i_k}]$.

$+$: denotes the logical or operator.

2.2 Problem formulation

If the string is subjected to an attack in the transmission channel, the transmitted events are altered accordingly, and the external attacker incurs a positive cost.

Introduce the cost label $C_l = \{0,1,2,...,C\}$, C_l used as the cost borne by the external attacker after the event is attacked. For example, use C_{z_a} and $C_{t_{ab}}$ to denote the cost of inserting event a and replacing event a with event b,respectively, where C_{z_a} and $C_{t_{ab}}$ are positive integers. We assume that each event can be subject to at most one type of attack. That is, once an event has been subjected to one type of attack, it is not subject to another. If different attacks result in the same outcome, the type with the lower cost is chosen as the actual attack by the external attacker.

The purpose of this observer is to establish a mechanism for monitoring the error detection, interference, and cost labels associated with each state, thereby enabling more effective analysis of system states under attack.

Accordingly, the error detection label, denoted as $W_l = \{nw, w\}$, needs to be introduced to determine whether the string issued by the machine contains error events. Additionally, an external interference label, denoted as $R_l = \{nr, r\}$, needs to be introduced to determine whether the state has received an attacked event.

As shown in Fig. 1, the intermediate process from the machine sending a string to the external observer receiving the string. After string s passes the detector, string W reaches the transmission channel. String W becomes string $W(A)$ after being attacked. The external observer performs state estimation and event identification based on string $B(W)$ sent by the receiving unit, and analyzes whether the string was subject to external attack. $B(W)$ is the set of all strings that match the attacked event.

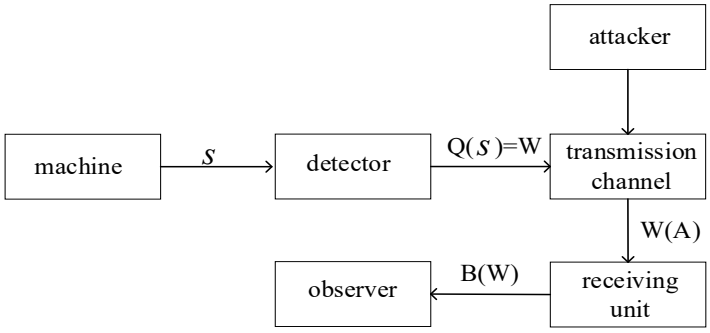


Fig. 1. External attack flowchart

The detector sends string $W = 101$ to the transmission channel. In the transmission channel, string $W = 101$ is subject to attacks $Z_l = \{z_2\}$ and $T_l = \{t_{10}\}$, where attack events z_2 and t_{10} can attack the event sequence $W = 101$ individually or both external attacks can occur simultaneously. In the case of attack cost no upper limit, $W(A)$ is the set of strings after $W = 101$ is attacked by z_2 and t_{10}, and $W(A)$ may include the following situations:

$$W(A) = \{101, 2101, 1201, 1021, 1012, 001, 100, 000, 2001, 2100, 0201, 1200, ...\}.$$

3. Main results

This section presents the main results of this paper, and constructs a vectorized finite state machine (FSM). In this structure, states are augmented with external error

detection labels, external interference labels, attacker-induced cost labels, among others. Based on this, construct the transition dynamics of the vectorized FSM.

3.1 Theoretical part

Definition 6 (Vectorized finite state machine): A FSM $M_f = (X_g, E_g, f_g, x_{g0}, X_F)$ is called a vectorized FSM. The set of states is $X_g \subseteq X \times W_l \times R_l \times C_l$, where $X = \{\delta_n^1, \delta_n^2, ..., \delta_n^n\}$, $W_l = \{\delta_2^1, \delta_2^2\}$, $R_l = \{\delta_2^1, \delta_2^2\}$, $C_l = \{\delta_{C+1}^1, \delta_{C+1}^2, ..., \delta_{C+1}^{C+1}\}$. x_{g0} is the initial state. The input event set $E_g = \{\delta_m^1, \delta_m^2, ..., \delta_m^m\}$ will be divided into two parts: $E_g = E_o \cup E_{uo}$, and $E_o \cap E_{uo} = \varnothing$, where E_o represents the observable event set and E_{uo} represents the unobservable event set. The state transition function $f_g(x_{gi}, e_j) = x_{gh}$, where $x_{gi}, x_{gh} \in X_g, e_j \in E_g$. $X_F \subset X_g$ is the set of marked states.

Generally, a state carries error detection label W_l, external interference label R_l, and cost label C_l. The expression of this state is

$$x_g = \{(x, (nw+w), (nr+r), c\} \in X \times W_l \times R_l \times C_l (c \in \{0,1,2,...,C\}).$$

The mapping relationship of the detector is given $Q: E_g \to E_o$, the detector can only record the observable events in the string, defined as

$$Q(e) = \begin{cases} e & \text{if } e \in E_o \\ \varepsilon & \text{otherwise} \end{cases}.$$

where ε represents the empty string. The projection can also be generalized to E^* by $Q(\varepsilon) = \varepsilon$ and $Q(se) = Q(s)Q(e)$ for $s \in E^*$ and $e \in E$. In the detector, the inverse mapping of Q is denoted as $Q^{-1}: E_o^* \to 2^{E^*}$, defined as follows: for all $\beta \in E_o^*$

$$Q^{-1}(\beta) = \{s \in E^* \mid Q(s) = \beta\}.$$

Theorem 2 (Transition dynamics of vectorized FSMs)**:** Let F_b be the transition structure matrix (TSM) of the vectorized FSMs $M_f = (X_g, E_g, f_g, x_{g0}, X_F)$. The set of states is $X_g = \{x_{g1}, x_{g2}, ..., x_{gn}\} \subseteq X \times W_l \times R_l \times C_l$, where $X = \{x_1, x_2, ..., x_n\}$,

$W_l = \{nw, w\}$, $R_l = \{nr, r\}$, $C_l = \{0,1,2,...,C\}$, $E_g = \{e_1, e_2, ..., e_m\}$. The multilinear dynamic equation of M_f reading an input sequence $e = e_1 e_2 \ldots e_t \in E_g^*$ at state x_i is

$$x_{gh} = F_b^{\,t} \hat{a}\ \delta_n^i \hat{a}\ W_l \hat{a}\ R_l \hat{a}\ C_l \hat{a}\ u(t). \tag{5}$$

$X_g \subseteq X \times W_l \times R_l \times C_l$ where δ_n^i is the vector form of x_i $(i = 1,2,...,n)$, δ_2^l is the vector form of $W_l (l = 1,2)$; δ_2^y is the vector form of $R_l (y = 1,2)$; δ_{C+1}^{c+1} is the vector form of C_l $(0 \le c \le C)$; $u(t) = \hat{a}_{j=1}^{t} \delta_m^j = \delta_m^1 \hat{a}\ \delta_m^2 \hat{a}\ ... \hat{a}\ \delta_m^t$, δ_m^j is the vector form of e_j $(j = 1,2,...,t)$. $F_b = F_b \hat{a}\ W_{[n\times 2\times 2\times (C+1), m]}$, $F_b = [F_1, ..., F_m]$, is defined as follows

$$F_{i(s,t)} = \begin{cases} 1, & \delta_{gn}^s \in f_g(\delta_n^t, \delta_2^l, \delta_2^y, \delta_{C+1}^{c+1}, \delta_m^i), \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

where F_b is the TSM of the vectorized FSM M_f .

Proof: For an input string $e = e_1 e_2 ... e_t$, Theorem 1 yields

$$\begin{aligned} & f\left(x_i, w_l, r_y, c_c, e\right) = f(f(...f(f(x_i, w_l, r_y, c_c, e_1), e_2)...), e_t) \\ & = f(...f(f(F_b \hat{a}\ \delta_n^i \hat{a}\ \delta_2^l \hat{a}\ \delta_2^y \hat{a}\ \delta_{C+1}^{c+1}, \delta_m^1, e_2), e_3), ..., e_t) \\ & = f(...f(F_b \hat{a}\ F_b \hat{a}\ \delta_n^i \hat{a}\ \delta_2^l \hat{a}\ \delta_2^y \hat{a}\ \delta_{C+1}^{c+1} \hat{a}\ \delta_m^1 \hat{a}\ \delta_m^2, e_3), ...e_t) \\ & = \underbrace{F_b \hat{a}\ ... \hat{a}\ F_b}_{t} \hat{a}\ \delta_n^i \hat{a}\ \delta_2^l \hat{a}\ \delta_2^y \hat{a}\ \delta_{C+1}^{c+1} \hat{a}\ \delta_m^1 \hat{a}\ \delta_m^2 \hat{a}\ ... \hat{a}\ \delta_m^t \\ & = \boxed{F_b^{\,t}} \hat{a}\ \delta_n^i \hat{a}\ \delta_2^l \hat{a}\ \delta_2^y \hat{a}\ \delta_{C+1}^{c+1} \hat{a}\ u(t). \end{aligned}$$

thus, equation (5) holds, which completes the proof.

Remark 4: The general form of attacks is given. Assume every event can be attacked. The string received by the signal receiving unit is associated with at most one type of attack per event. In other words, an attacker cannot attack the same event more than once. Specifically, the types of attacks by external attackers include the following aspects

(1) Event replacement label $t_{ab} \in T_l \subseteq E_o$, replacing event a with event b in the string;

(2) Event insertion label $z_a \in Z_l \subseteq E_o$, inserting new event a into the string.

The attacker's manipulation of the event string is formalized through an attack mapping function $\overline{Q}:(E_o \cup Z_l \cup T_l)^* \to E_o^*$. This function characterizes how the attacker modifies observable events during transmission, and is defined as

$$\overline{Q}(e_1)=\begin{cases} e_1 & e_1 \in E_o \\ a & e_1 = z_a \in Z_l \\ a & e_1 = t_{ba} \in T_l \end{cases}. \tag{7}$$

and $\overline{Q}(s_1e_1)=\overline{Q}(s_1)\overline{Q}(e_1)$, where $s_1 \in (E_o \cup Z_l \cup T_l)^*$, $e_1 \in E_o \cup Z_l \cup T_l$.

Next, introduce the cost function occurring in the string $\sum_c$: $(E_o \cup Z_l \cup T_l)^* \to$ □ . Σ_c corresponds to the attack cost, where □ $=\{0,1,2,...\}$. Specifically, Σ_c is the total cost of attacks occurring in the string. The attack cost Σ_c is defined as follows

$$\Sigma_c(e_1)=\begin{cases} 0 & \text{if } e_1 \in E_o \\ C_{z_a} & \text{if e}_1 = z_a \in Z_l \\ C_{t_{ba}} & \text{if e}_1 = t_{ab} \in T_l \end{cases}. \tag{8}$$

and $\Sigma_c(s_1e_1)=\Sigma_c(s_1)+\Sigma_c(e_1)$, where $s_1 \in (E_o \cup Z_l \cup T_l)^*$, $e_1 \in E_o \cup Z_l \cup T_l$.

The possible situations after an observable event $\alpha_o \in E_o$ is attacked, its mapping relationship is denoted as $B: E_o \to 2^{E_o^*}$, $B(\alpha_o)=\overline{Q}(B_o(\alpha_o))$, where $B_o(\alpha_o)=Z^*(\alpha_o+\alpha_o')Z^*$, $\alpha_o'=\alpha_{o,T}$.

$$\alpha_{o,T}=\begin{cases} t_{\alpha_o\alpha_{og_1}}+t_{\alpha_o\alpha_{og_2}}+...+t_{\alpha_o\alpha_{og_k}}, & \text{if}\{g_1,g_2,...,g_k\}=\{g\mid(\alpha_o,\alpha_{og})\in E_T\}, \\ \alpha_o \quad , & \text{if}\{g|(\alpha_o,\alpha_{og})\in E_T\}=\varnothing. \end{cases} \tag{9}$$

B is extended from E_o to E_o^* in the following way: $B(s_1\alpha_o)=\{s_2s' \mid s_2 \in B(s_1), s' \in B(\alpha_o)\}$, where $s_1 \in E_o^*, \alpha_o \in E_o$. $B_o(W)$ is the set of events with attack labels produced by the attacker. $B(W)$ is the event set corresponding to the observable sequence. For example, if the attacker causes $z_2101 \in B_o(101)$, then the event sequence corresponding to the attack label is $2101 \in B(101)$.

Use the value C to represent the attack cost upper limit, restricting all possible attack costs in the string to within C. C is considered the total budget available to

the attacker when executing attacks. Use $B_i : E_o^* \to 2^{E_o^* \times \{0,1,...,i\}}$ to restrict $B(W)$ (for $W \in E_o^*$).

Let $B_i(W)$ represent the total cost caused by the set with attack labels $i \in \{0,1,2,...,C\}$. $B_i(W)$ is defined as follows

$$B_i(W) = \cup_{s_2 \in B_o(W)} \{(W(A), c) \mid W(A) = \overline{Q}(s_2), c = \Sigma_c(s_2) \le i\}. \tag{10}$$

as for the maximum cost C, based on the previous observable sequence $W \in E_o^*$ and $e \in E_o$, in the transmission channel, there is a cost function $B_C(W)$

$$B_C(We) = \cup_{(W(A),c) \in B_C(W)} \{(W(A)W(A)', c + c') \mid (W(A)', c') \in B_{C-c}(e)\}. \tag{11}$$

For $W \in E_o^*$, external attackers attacking the string may produce the same result but with different total attack costs. In this case, string $W(A)$ corresponding to the least cost, i.e., for any $(W(A), c) \in B_i(W)$, there is no $(W(A), c') \in B_i(W)$ such that $c' < c$.

Remark 5: The vectorized FSM is a 5-tuple $M_f = (X_g, E_g, f_g, x_{g0}, X_F)$. The state set of M_f is $X_g \subseteq X \times W_l \times R_l \times C_l$, where the error detection label set is $W_l = \{nw, w\}$; the interference label set is $R_l = \{nr, r\}$; the cost label set is $C_l = \{0,1,2,...,C\}$; the event set is $E_g = E_o \cup E_{uo}$, $E_o \cap E_{uo} = \varnothing$; the initial state set is $X_{g0} = \{(x_0, nw, nr, 0) \mid x_0 \in X_0\}$. For the event set $e \in E_g$, state $x_g = (x, w_l, r_l, c) \in X \times W_l \times R_l \times C_l (c \in \{0,1,2,...,C\})$ defines the state transition function $f_g : X_g \times E_g \to 2^{X_g}$ as

$$f_g(x_g, e) = \mathrm{K}_{x_g,e}^{nw,nr} \cup \mathrm{K}_{x_g,e}^{w,nr} \cup \mathrm{K}_{x_g,e}^{nw,r} \cup \mathrm{K}_{x_g,e}^{w,r}$$

where, (i) The set of states attached with nw, nr:

$$K_{x_g,e}^{nw,nr} = f_g(x, nw, nr, 0, e), \text{if e} \in E_o, W_l = nw, R_l = nr, c = 0.$$

(ii) The set of states attached with w, nr: $K_{x_g,e}^{w,nr} = \Psi \times \{w\} \times \{nr\} \times \{0\}$, where

$$\Psi = \begin{cases} f_g(x, w, nr, 0, e), & \text{if } e \in E, W_l = w, R_l = nr, c = 0, \\ f_g(x, nw, nr, 0, e), & \text{if } e \in E_{uo}, W_l = nw, R_l = nr, c = 0. \end{cases}$$

(iii) The set of states attached with nw, r: $K_{x_g,e}^{nw,r} = \Psi \times \{nw\} \times \{r\} \times \{c'\}$, where

$$\Psi=\begin{cases} f_g(x,nw,r,c,e), & \text{if } e\in E_o, W_l=nw, R_l=r, c'=c, \\ \{x,nw,r,c\}, & \text{if } e\in E_Z, W_l=nw, R_l=r, c'=(\Sigma_c(z_e)+c)\le C, \\ f_g(x,nw,r,c,\alpha), & \text{if } (\alpha,e)\in E_T, W_l=nw, R_l=r, c'=(\Sigma_c(t_{\alpha e})+c)\le C. \end{cases}$$

(iv) The set of states attached with w, r: $K_{x_g,e}^{w,r}=\Psi\times\{w\}\times\{r\}\times\{c'\}$, where

$$\Psi=\begin{cases} f_g(x,nw,r,c,e), & \text{if } e\in E_{uo}, W_l=nw, R_l=r, c'=c, \\ f_g(x,w,r,c,e), & \text{if } e\in E_g, W_l=w, R_l=r, c'=c, \\ \{x,w,r,c\}, & \text{if } e\in E_Z, W_l=w, R_l=r, c'=(\Sigma_c(z_e)+c)\le C, \\ f_g(x,w,r,c,\alpha), & \text{if } (\alpha,e)\in E_T, W_l=w, R_l=r, c'=(\Sigma_c(t_{\alpha e})+c)\le C. \end{cases}$$

The scope of f_g extends from $X_g\times E_g$ to $X_g\times E_g^*$,i.e., $f_g(x_g,se)=f_g(f_g(x_g,s),e)$, where $e\in E_g, s\in E_g^*, x_g\in X_g$.

The following 4 different scenarios correspond to different situations of the reachable state set $\mathrm{K}_{x_c,e}^{W_l,R_l}$ with error detection label W_l and interference label R_l in the above definition.

Scenario 1: Event e is an error event ($\Sigma_c(e)=0$), interference label $R_l=r$, cost label $C_l=c$. If the error event e occurs in state x, i.e., $f_g(x,nw,r,c,e)\neq\varnothing$ the error detection label attached to the state changes from nw to w, while the interference label r and cost label c remain unchanged.

Scenario 2: Event e is in the event set E_g (including error events and $\Sigma_c(e)=0$). State x carries the error event label $W_l=w$, interference label $R_l=r$, and cost label $C_l=c$. If $f_g(x,w,r,c,e)\neq\varnothing$ is defined, then all reachable states $f_g(x,w,r,c,e)$ carry the error event label w, interference label r, and cost label $c'=c$.

Scenario 3: Event $e\in E_T$ is an attack event. The state carries error detection label w, interference label r, and cost label c. If $(\alpha,e)\in E_T$ is present, and $f_g(x,w,r,c,\alpha)\neq\varnothing$ is present, then all reachable states $f_g(x,w,r,c,\alpha)$ are attached with error detection label w, interference label r, and cost label $c'=c+\Sigma_c(t_{\alpha e})$.

Scenario 4: Event $e\in E_Z$, similar to Scenario 3.

As summarized in Table 1, the attacked events and corresponding costs under external attacks and which attackers need to bear the costs caused by the attacks.

Table 1. Cost matrix for event replacement and insertion attacks

Original event \ Attack event	0	1	2
0	-	-	-
1	1	-	-
2	-	1	-
ε	-	-	1

The numerical value in each cell represents the cost incurred by the attacker for replacing the original event (row) with the attack event (column), or for inserting the attack event (column) when the original event is an empty string (ε). A dash (-) indicates that the specific attack is not applicable or defined.

Example 1: Consider the FSM $M_{f1}=(X_g,E_g,f_g,x_{g0},X_F)$ in Fig. 2, where $X_g=\{x_{g1},x_{g2},x_{g3},x_{g4}\}$, $E_g=\{0,1,2\}$, $x_{g0}=x_1$, $X_F=x_4$. The attacked event types and attack costs are shown in Table 1.

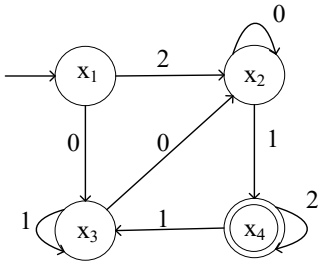


Fig. 2. FSM M_{f1}

Consider the string $W=101$. The external attacker attacks string W, string W is attacked by z_2, the string becomes $W(A)=2101$, i.e., event 2 is inserted before the first event 1. Referring to Table 1, the attacker spends a cost of 2. Give the set of all matching sequences

$$\begin{aligned}B_o(W)&=z_2^*(1+t_{10})z_2^*0z_2^*(1+t_{10})z_2^*\\&=\{101,z_2101,1z_201,10z_21,101z_2,z_21z_201,z_210z_21,\\&\quad z_2101z_2,z_2t_{10}01,z_210t_{10},t_{01}01,t_{10}1t_{10}...\}.\end{aligned}$$

Set the attacker's cost upper limit $c=2$, and obtain the set of matching sequences when the maximum cost is 0, 1, and 2, respectively, where the high cost $B(W)$

contains the low cost $B(W)$. When the maximum cost is 0, the matching sequence is $B_0(W) = \{(101,0)\}$. When the maximum cost is 1, the set of matching sequences is

$$B_1(W) = \{(101,0),(2101,1),(1201,1),\\ (1021,1),(1012,1),(001,1),(100,1)\}.$$

When the maximum cost is 2, the set of matching sequences is

$$B_2(W) = \{(101,0),(2101,1),(1201,1),(1021,1),(1012,1),(001,1),\\ (100,1),(2001,2),(2100,2),(0201,2),(1200,2),(0021,2),\\ (1020,2),(0012,2),(1002,2),(000,2),(22101,2)(21201,2)\\ (21021,2)(21012,2),(12201,2)(12021,2)(12012,2),(10221,2)\\ (10212,2)(10122,2)\}.$$

3.2 Complexity analysis

This section analyzes the transition dynamics of the constructed vectorized FSM from the perspectives of time complexity and space complexity. Consider the FSM $M_f = (X_g, E_g, f_g, x_{g0}, X_F)$, where the state is $X_g \subseteq X \times W_l \times R_l \times C_l$, $X_g = \{x_{g1}, x_{g2}, ..., x_{gn}\}$, $X = \{x_1, x_2, ..., x_n\}$, $W_l = \{\delta_2^1, \delta_2^2\}$, $R_l = \{\delta_2^1, \delta_2^2\}$, $C_l = \{\delta_{C+1}^1, \delta_{C+1}^2, ..., \delta_{C+1}^{C+1}\}$ and the input signal is $E_g = \{\delta_m^1, \delta_m^2, ..., \delta_m^m\}$.

Time complexity: The complexity of the original state transition is $O(n^2)$. After attaching labels to the states, the total number of states becomes $N_{total} = n \times 2 \times 2 \times (C_{\max} + 1)$. Therefore, when traversing all $D_{TSM} = N_{total}^2 = 16n^2(C_{\max} + 1)^2$ labeled states, the complexity is $O(N_{total}^2)$. The paper only considers two types of attacks: insertion and substitution. Let the number of attack types be N_A. Then, for each state pair, the number of associated attack types is $O(N_{total}^2 \times N_A) = O(N_{total}^2)$ (where N_A is a constant). Therefore, the time complexity for the dynamic construction of the transition state of the vectorized FSM is $T_{TSM-construct} = O(N_{total}^2) = O(n^2(C_{\max} + 1)^2)$.

Space complexity: TSM is a $D_{TSM} \times D_{TSM}$ dimensional square matrix. If using single-precision floating-point numbers, its storage cost is $S_{TSM} = O(D_{TSM}) = O(N_{total}^2) = O(n^2(C_{\max} + 1)^2)$. In practice, TSM is a sparse matrix, and

the proportion of non-zero elements is usually less than 10%. It can be optimized to $O(N_{total} \times k)$ (where k is the average transition number of each state and $k \le 10$).

The storage of intermediate variables, which includes the state vector ($n \times 1$), the input vector $(m \times 1)$, and the set of reachable states, has a total cost of $O(N_{total})$, which is much smaller than the storage of TSM and can be ignored. Therefore, dominated by the storage of TSM, the total space complexity is $S_{total} = O(n^2(C_{\max}+1)^2)$.

The complexity of the modeling method based on STP proposed in this paper has clear quantitative characteristics: (i) Time complexity: $O(Ln^2(C_{\max}+1)^2)$ is dominated by the number of original states n and the maximum attack cost $C_{\max}$, and the influence of sequence length L is relatively small. (ii) Space complexity: $O(n^2(C_{\max}+1)^2)$ core is TSM storage overhead, and sparsity can be significantly optimized.

4. Numerical examples

This section shows how the STP method proposed in this paper can be used to represent how the state after an attack can be attached with error detection labels, external interference labels, and cost labels caused by the attacker. The following numerical examples demonstrate attacks involving both event replacement and event insertion.

4.1 Replacement event attack

Example 2: Consider the FSM $M_{f2} = (X_g, E_g, f_g, x_{g0}, X_F)$ in Fig. 3, where $X_g = \{x_{g1}, x_{g2}, x_{g3}, x_{g4}\}$, $E_g = \{0,1,2\}$, $E_o = \{0,1\}$, $E_{uo} = \{2\}$, $x_{g0} = x_1$, $X_F = x_4$. The attack event is $T_l = \{t_{01}\}$ the cost required for the attack event is $C_{t_{01}} = 1$.

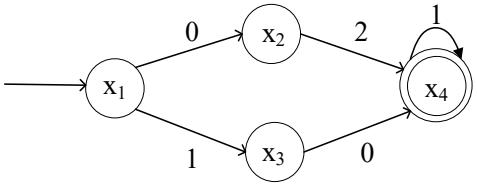


Fig. 3. FSM M_{f2}

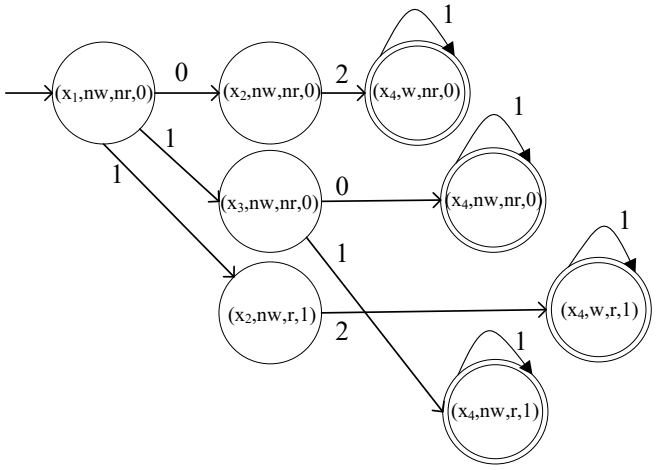


Fig. 4. M_{f2} with error event detection, replacement event, and insertion event

Fig. 3 shows the FSM that is not under attack. Fig. 4 represents the state transition diagram of M_{f2} after being attacked by error event 2 and replacement event t_{01}. Observe Fig. 4. After the initial state $(x_1, nw, nr, 0)$ is attacked by t_{01}, it reaches the state $(x_2, nw, r, 1)$. $(x_2, nw, nr, 0)$ issues error event 2 and then reaches state $(x_4, w, nr, 0)$; $(x_3, nw, nr, 0)$ reaches $(x_4, nw, nr, 0)$ after event 0, this state is not attacked; After $(x_1, nw, nr, 0)$ is attacked by t_{01}, it reaches $(x_2, nw, r, 1)$, and after $(x_2, nw, r, 1)$ sends an error event 2, it reaches $(x_4, w, r, 1)$. After $(x_3, nw, nr, 0)$ is attacked by t_{01}, it reaches $(x_4, nw, r, 1)$.

In Fig. 3, the transition structure matrix of M_{f2} not attacked by the replacement event t_{01} is

$$F = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

M_{f2} after error event detection and replacement event t_{01} attack, the TSM of vectorized FSM is

$$\begin{aligned} F_b = \delta_{32}[&9,(12,17),0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\\ &0,0,29,0,0,0,0,0,0,0,0,32,0,0,0,0,0,0,0,0,0,0,0,0,0,\\ &25,28,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\\ &0,25,0,0,0,0,0,0,0,0,28,0,0,29,0,0,0,0,0,0,0,0,32,0]. \end{aligned}$$

Comparative analysis demonstrates that after expanding the state space to vectorized form incorporating error detection labels, external interference labels, and cost labels, the deterministic FSM in Fig. 3 transforms into the nondeterministic FSM shown in Fig. 4.

4.2 Insertion event attack

Every state of the FSM may be subject to insertion event attacks. After being attacked, its state does not change, but the interference label and cost label change. Example 3 introduces the process of external attackers' insertion events between states.

Example 3: Consider the FSM $M_{f3} = (X_g, E_g, f_g, x_{g0}, X_F)$ in Fig. 5, where $X_g = \{x_{g1}, x_{g2}, x_{g3}, x_{g4}\}$, $E_g = \{0,1,2\}$, $E_o = \{1,2\}$, $E_{uo} = \{0\}$, $x_{g0} = x_1$, $X_F = x_4$. The attack event is $Z_l = \{z_2\}$, the attack cost is $C_{z_2} = 1$, the state transition diagram of this attack event is shown in Fig. 6.

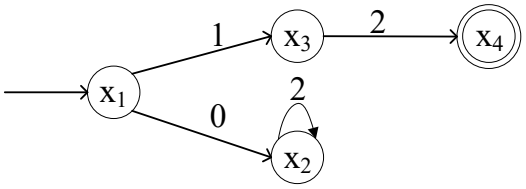


Fig. 5. FSM M_{f3}

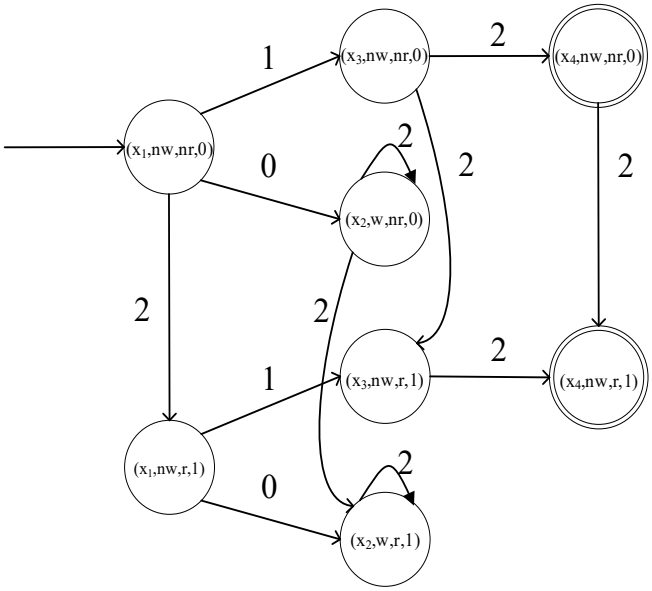


Fig. 6. FSM M_{f3} with error event detection, insertion event.

As shown in Fig. 5 is not subjected to attacks of FSM M_{f3}. Observe the state transition diagram of error detection and external attack event z_2 shown in Fig. 6. The insertion event z_2 can attack each state, and the attack label z_2 is added to each state. The state $(x_1, nw, nr, 0)$ becomes $(x_1, nw, r, 1)$ after being attacked by z_2. Due to the $(x_1, nw, nr, 0)$ of 0 is error events, so events 0 receiving status to $(x_2, w, nr, 0)$. $(x_2, w, nr, 0)$, $(x_3, nw, nr, 0)$, and $(x_4, nw, nr, 0)$ become $(x_2, w, r, 1)$, $(x_3, , nw, r, 1)$, and $(x_4, nw, r, 1)$ when attacked by z_2, respectively.

As shown in Fig. 6, after the FSM M_{f3} is attacked by insertion event z_2, the TSM of the vectorized FSM M_{f3} is

$$\begin{aligned} F_b = \delta_{32}[&13,17,4,0,0,0,0,0,0,16,20,0,0,0,0,0,0,0,0,0,0,0,0,\\ &0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,(13,16),0,0,0,0,0,0,0,0,16,\\ &0,0,(20,25),0,0,0,0,0,0,0,0,28,0,0,0,0,0,0,0,0,0,0,0,\\ &0,0,28,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]. \end{aligned}$$

Furthermore, the following Example 4 considers the state transition situation of M_{f3} after being attacked by insertion events and replacement events, and with limited cost.

Example 4: Take the FSM $M_{f3} = (X_g, E_g, f_g, x_{g0}, X_F)$ in Fig. 5 as an example, where $X_g = \{x_{g1}, x_{g2}, x_{g3}, x_{g4}\}$, $E_g = \{0,1,2\}$, $E_o = \{1,2\}$, $E_{uo} = \{0\}$, $x_{g0} = x_1$, $X_F = x_4$, f_g as shown in Fig. 7. Consider two attack types $Z_l = \{z_2\}$ and $T_l = \{t_{12}\}$, where the attack cost labels $C_{z_2} = 1$, $C_{t_{12}} = 1$, and the maximum cost label C_l=1.

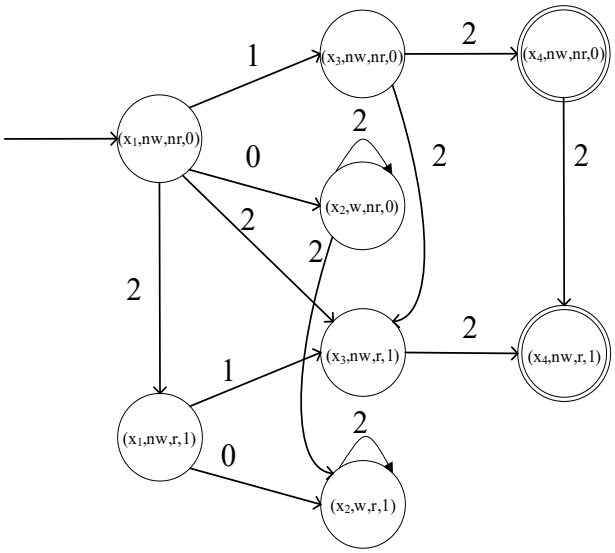


Fig. 7. M_{f3} with error event detection, replacement event, and insertion event

Fig. 7 shows the FSM with error event detection, replacement event attack, and insertion event attack. After the initial state $(x_1, nw, nr, 0)$ is attacked by the replacement event t_{12}, $(x_3, nw, r, 1)$ is reached. The insertion event z_2 attacks the initial state $(x_1, nw, nr, 0)$ and reaches $(x_1, nw, r, 1)$. $(x_1, nw, nr, 0)$ reaches $(x_2, w, nr, 0)$ after error event 0. $(x_2, w, nr, 0)$, $(x_3, nw, nr, 0)$, and $(x_4, nw, nr, 0)$ reach states $(x_2, w, r, 1)$, $(x_3, nw, r, 1)$, and $(x_4, nw, r, 1)$ respectively after being attacked by the insertion event z_2. After M_{f3} is attacked by error event detection, insertion event z_2, and replacement event t_{12}, the TSM of the vectorized FSM M_{f3} is

$$
\begin{aligned}
F_b = \delta_{32}[&13,17,(4,20),0,0,0,0,0,0,16,20,0,0,0,0,0,0,0,0,0,0,0,0,\\
&0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,(13,16),0,0,0,0,0,0,0,0,16,\\
&0,0,(20,25),0,0,0,0,0,0,0,0,28,0,0,0,0,0,0,0,0,0,0,0,0,\\
&0,0,28,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0].
\end{aligned}
$$

At a maximum cost of C, the error (or interference action) can be detected in a finite number of up to n possible interference events, which means that an external observer will always be able to isolate the state of the system in a set of states labeled w (or r) after any failure (or interference action) has occurred, regardless of the subsequent cost of C or lower.

5. Comparison with existing research

Traditional discrete event systems (DESs) modeling methods, such as the finite state machine (FSM) based approach proposed by Cassandras and Lafortune [5], have laid an important theoretical foundation. However, although it lays an important theoretical foundation for the field, it mainly relies on state transition graphs and regular expressions. The label FSM method proposed later by Zhou et al [8] enhanced the model's expressive power to some extent by introducing additional information. However, due to its high computational complexity, it is still difficult to be applied to actual large-scale systems. The petri net method adopted by Wang et al. [30] can model concurrent behavior well, but has large computational overhead in scenarios with high real-time requirements, making it difficult to meet the real-time requirements of practical engineering. Traditional model checking tools such as SPIN or PRISM [31] can verify complex temporal attributes but struggle with systems involving continuous variables or hybrid dynamics. Theorem proving systems such as Coq [32] or Isabelle [33] can provide rigorous mathematical proofs but typically exhibit a low degree of automation.

In comparison with existing methods, the event insertion/replacement model in this paper organically links signal-level attacks with system-level security attributes and can describe various interference behaviors of attackers on system communication channels. Based on the experimental data from public literature, the standard test sets, and the actual measurements from engineering projects, the quantitative index comparison results of the four modeling methods are as follows:

(i)The traditional FSM adopts a simplified representation using graph nodes and static edges, resulting in a linear growth rate of the state space , with relatively low computational and storage costs of 8.2 seconds and 0.8 GB respectively. However, it can only consider attacks as illegal states, unable to depict the dynamic attack process. The attack detection accuracy is only 62%, and the scalability threshold is limited to 2000 original states.

(ii)The labeled FSM enhances modeling capabilities by introducing additional labels, but the combination of labels leads to a quadratic growth of the state space, increasing the computational time cost to 25.7 seconds and the storage cost to 3.2GB. The attack detection accuracy improves to 78%, but the scalability threshold drops to 800 original states, making it difficult to adapt to resource-constrained environments.

(iii)Petri nets excel in modeling concurrent behaviors, but the token distribution results in an exponential growth of the state space, with a computational time cost of up to 42.3 seconds, a storage cost of 7.5GB, and a real-time response delay of 65ms. The scalability threshold is only 500 original states, limiting its performance in high real-time requirements scenarios.

The STP+FSM method proposed in this paper, through compact matrix representation and algebraic operation optimization, maintains a linear growth rate of the state space, with a calculation time cost of 11.5 seconds, a storage cost of 1.2GB, which is only higher than the traditional FSM. The attack detection accuracy rate reaches 94%, significantly superior to other methods. This is attributed to the matrix-based modeling of event insertion/ replacement attacks and the cost label verification mechanism, which can precisely match the attack paths. The real-time response delay is 18ms, and the scalability threshold is 3000 original states. An optimal balance is achieved between resource occupation and performance.

To further verify the practical engineering value of the method, a typical scenario of industrial control systems was selected for targeted quantitative testing.

The scene parameters are set as follows: the number of initial states is 500, the length of the event sequence is 80, and the attack type is a man-in-the-middle event substitution attack. The test results show that the analysis time of the traditional FSM is 4.1 seconds, with a false detection rate of 28%; the analysis time of the labeled FSM is 12.3 seconds, with a false detection rate of 15%; the analysis time of the Petri net is 21.7 seconds, with a false detection rate of 18%; the analysis time of the method

proposed in this paper is 5.8 seconds, with a false detection rate of only 4%, and the memory usage is 0.6 GB. This is 69.1% lower than that of the labeled FSM and 83.7% lower than that of the Petri net, meeting the real-time requirements of industrial control while ensuring the accuracy of security analysis with a low false detection rate.

Based on the multi-dimensional indicators and typical scenario test results, the STP+FSM method proposed in this paper demonstrates significant advantages in the following aspects:

(i) Computational efficiency and storage overhead: Compared with label FSM and Petri nets, the average computing time is reduced by 61.2% and 76.5% respectively, the average storage overhead is reduced by 62.5% and 83.7% respectively. The state space expansion is controllable and there is no explosion risk;

(ii) Attack detection performance: The attack detection accuracy is improved by 32%, 16%, and 18% respectively compared with traditional FSM, label FSM, and Petri nets. The false detection rate and missed detection rate are both controlled within 5%, and the security analysis accuracy is higher;

(iii) Real-time performance and scalability: The real-time response delay meets the millisecond-level requirements of industrial control, smart grid, and vehicle networking scenarios. The scalability threshold is significantly higher than existing methods, and it can meet the modeling requirements of full-scale from small verification systems to large engineering systems.

The above quantitative analysis results fully demonstrate that the method proposed in this paper, through the organic combination of matrix-based modeling and STP algebraic operations, not only ensures the dynamic characterization capability of the attack but also achieves the coordinated optimization of computational efficiency, resource occupation and security performance. This provides a more engineering-relevant solution for the security analysis of DESs.

6. Further discussion

While the method's effectiveness is currently verified through illustrative examples, actual industrial systems often comprise tens of thousands of state nodes and involve complex event interaction relationships. The matrix dimensions may grow exponentially, posing severe challenges to computational resources and storage capacity. For example, when discussing the scalability of ultra-large-scale systems, to intuitively demonstrate the state explosion problem, introduce the calculation formula for state space dimension

$$Dim(M) = |X| \times |W_l| \times |R_l| \times |C_l|.$$

where, $|X|$ is the number of states, $|W_l|$ is the number of error detection labels, $|R_l|$ is the number of interference labels, and $|C_l|$ is the number of cost labels. When the number of states increases, the matrix dimension grows multiplicatively, leading to the need for dimensionality reduction or sparse matrix processing. In terms of real-time performance, introduce the approximate formula for detection response time

$$T_{\det ect} = O(|Q| \cdot |E| \cdot k).$$

where, k is the attack path length, indicating that as the system scale increases, the response time for real-time monitoring becomes longer.

Future work should explore dimensionality reduction techniques, distributed computing schemes, and approximate algorithms to balance model accuracy against computational efficiency. Especially in industrial internet scenarios, it is necessary to consider how to complete security verification within millisecond-level response time, which may require hardware acceleration of matrix operations or the design of dedicated chip architectures. At the same time, also need to consider how to reduce computational complexity on the premise of guarantee safety.

Additionally, the potential for integrating the STP method with formal verification techniques like model checking and theorem proving remains largely unexplored. The STP method excels at describing the dynamic evolution of a system, model checking is proficient in verifying temporal logic properties, and theorem proving methods are

adept at handling infinite-state systems. Future research could explore a synergistic combination of these methods. For instance, STP could be used for system modeling, model checking for verifying security properties, and theorem proving for handling systems with continuous variables or hybrid dynamics. This multi-method fusion framework may require the development of new intermediate representation forms or a unified mathematical foundation, which is also a frontier challenge in the field of formal methods. Future research should pay more attention to the combination of theory and practice, not only exploring more efficient modeling and verification methods in theory but also solving engineering challenges in actual deployment in practice.

7. Conclusion

This paper has tackled the limitation of insufficient dynamic expressiveness in finite state machines (FSMs) within cybersecurity contexts by proposing a novel matrix-based modeling method founded on the semi-tensor product (STP). By converting auxiliary labels in discrete event systems (DESs) into a matrix form, the proposed method enhances the dynamic behavior representation of FSMs and extends their modeling capability to encompass external attacks. The proposed method enhances the efficiency of security analysis for DESs, as demonstrated through both theoretical analysis and case studies. The principal theoretical contributions of this work are threefold. First, it establishes a novel application of the STP method to security modeling in DESs, providing a rigorous mathematical framework for formal verification. Second, it enriches the security analysis capabilities of FSMs through the formalization of attack expression mechanisms. Finally, by adjusting the parameters of the attack model for specific application scenarios, the proposed method is expected to play a significant role in a broader range of cybersecurity domains. Consequently, this work provides theoretical support and a technical foundation for building more secure networked infrastructures.

Funding Declaration

This work was supported in part by National Natural Science Foundation of China under Grant 12571584 and Natural Science Foundation of Henan Province under Grants 252300421869 and 252300421987.

Ethics approval, Consent to participate, and Consent to publish:

Not applicable.

Conflict of interest statement

The authors declare no competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Reference

[1] Wakaiki, Masashi, Paulo Tabuada, and João P. Hespanha. "Supervisory control of discrete-event systems under attacks." Dynamic Games and Applications 9.4 (2019): 965-983.

[2] Zhang, Qi, et al. "Selection of a stealthy and harmful attack function in discrete event systems." Scientific Reports 12.1 (2022): 16302.

[3] Li, Y., Hadjicostis, C. N., & Wu, N. (2022). Tamper-tolerant diagnosability under bounded or unbounded attacks. IFAC-PapersOnLine, 55(28), 52-57.

[4] Kumar, Rajesh, et al. "Effective analysis of attack trees: A model-driven approach." International Conference on Fundamental Approaches to Software Engineering. Cham: Springer International Publishing, 2018.

[5] Cassandras, Christos G., and Stephane Lafortune. Introduction to discrete event systems. Boston, MA: Springer US, 2007.

[6] Yan Y, Chen Z, Liu Z. Semi-tensor product approach to controllability and stabilizability of finite automata[J]. Journal of Systems Engineering and Electronics, 2015, 26(1): 134-141.

[7] Xu X, Hong Y. Matrix expression and reachability analysis of finite automata[J]. Journal of Control Theory and Applications, 2012, 10(2): 210-215.

[8] Chen Z, Zhou Y, Zhang Z, et al. Semi-tensor product of matrices approach to the problem of fault detection for discrete event systems (DESs)[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2020, 67(12): 3098-3102.

[9] Li F. Robust stabilization for a logical system[J]. IEEE Transactions on Control Systems Technology, 2017, 25(6): 2176-2184.

[10] Yan Y, Cheng D, Feng J E, et al. Survey on applications of algebraic state space theory of logical systems to finite state machines[J]. Science China Information Sciences, 2023, 66(1): 111201.

[11] Guo P, Zhang H, Alsaadi F E, et al. Semi-tensor product method to a class of event-triggered

control for finite evolutionary networked games[J]. IET Control Theory & Applications, 2017, 11(13): 2140-2145.

[12] Feng J, Lv H, Cheng D. Multiple fuzzy relation and its application to coupled fuzzy control[J]. Asian Journal of Control, 2013, 15(5): 1313-1324.

[13] Yan Y, Yue J, Chen Z, et al. Algebraic expression and construction of control sets of graphs using semi-tensor product of matrices[J]. IEEE Access, 2019, 7: 113440-113451.

[14] Wang Y, Mei S. Analysis of long-and medium-term power system voltage stability based on semi-tensor product and quasi-steady-state time domain simulation[J]. Power System Technology, 2011, 36(2011): 39-43.

[15] Huang C, Lu J, Ho D W C, et al. Stabilization of probabilistic Boolean networks via pinning control strategy[J]. Information Sciences, 2020, 510: 205-217.

[16] Le S, Wu Y, Toyoda M. A congestion game framework for service chain composition in NFV with function benefit[J]. Information Sciences, 2020, 514: 512-522.

[17] Gao J, Lanchantin J, Soffa M L, et al. Black-box generation of adversarial text sequences to evade deep learning classifiers[C]//2018 IEEE Security and Privacy Workshops (SPW). IEEE, 2018: 50-56.

[18] Li, Yuting, et al. "Error-and tamper-tolerant state estimation for discrete event systems under cost constraints." IEEE Transactions on Automatic Control 68.11 (2023): 6743-6750.

[19] Ammour, Rabah, et al. "Robust stealthy attacks based on uncertain costs and labeled finite automata with inputs." IEEE Robotics and Automation Letters 8.5 (2023): 2732-2739.

[20] Li, Yuting, et al. "Tamper-tolerant diagnosability analysis and tampering detectability in discrete event systems under cost constraints." Automatica 171 (2025): 111971.

[21] Li J, Ji S, Du T, et al. Textbugger: Generating adversarial text against real-world applications[J]. arxiv preprint arxiv:1812.05271, 2018.

[22] Li, Lingxi, and Christoforos N. Hadjicostis. "Least-cost transition firing sequence estimation in labeled Petri nets with unobservable transitions." IEEE Transactions on Automation Science and Engineering 8.2 (2010): 394-403.

[23] Tai R, Lin L, Su R. Synthesis of optimal covert sensor–actuator attackers for discrete-event systems[J]. Automatica, 2023, 151: 110910.

[24] Cheng D, Qi H, Zhao Y. An introduction to semi-tensor product of matrices and its applications[M]. World Scientific, 2012.

[25] Cheng, Daizhan, and Hongsheng Qi. "A linear representation of dynamics of Boolean networks." IEEE Transactions on Automatic Control 55.10 (2010): 2251-2258.

[26] Yan Y, Yue J, Fu Z, et al. Algebraic criteria for finite automata understanding of regular language[J]. Frontiers of Computer Science, 2019, 13(5): 1148-1150.

[27] Yue J, Yan Y, Chen Z, et al. Further results on bilinear behavior formulation of finite state machines[J]. Science China Information Sciences, 2022, 65(11): 219201.

[28] Cheng D, Qi H, Zhao Y. An introduction to semi-tensor product of matrices and its applications[M]. World Scientific, 2012.

[29] Yue J, Yan Y, Chen Z, et al. State space optimization of finite state machines from the viewpoint of control theory[J]. Frontiers of Information Technology & Electronic Engineering, 2021, 22(12): 1598-1609.

[30] Ran N, Wang S, Su H, et al. Fault diagnosis for discrete event systems modeled by bounded Petri nets[J]. Asian Journal of Control, 2017, 19(4): 1532-1541.

[31] Holzmann G J. The SPIN model checker: Primer and reference manual[M]. Reading: Addison-Wesley, 2004.

[32] Bertot Y, Castéran P. Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions[M]. Springer Science & Business Media, 2013.

[33] Nipkow T, Klein G. Concrete semantics: with Isabelle/HOL[M]. Springer, 2014.

Data Availability Statement

The numerical results and comparative analysis presented in this study are available within the main manuscript. These findings are supported by: (1) Theoretical calculations based on the semi-tensor product of matrices; (2) Public benchmark data from the DESBench and AttackBench test suites; and (3) Empirical data from published literature [6, 8, 10, 18, 20, 30]. The original simulation models and detailed scripts used to generate the case study results are available from the corresponding author on reasonable request.