mathematics

MDPI

Article

State-Based Differential Privacy Verification and Enforcement for Probabilistic Automata

Yuanxiu Teng, Zhiwu Li, Li Yin * and Naiqi Wu

Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau SAR, China; 2009853DMI30001@student.must.edu.mo (Y.T.); zwli@must.edu.mo (Z.L.); nqwu@must.edu.mo (N.W.)
* Correspondence: liyin@must.edu.mo

Abstract: Roughly speaking, differential privacy is a privacy-preserving strategy that guarantees attackers to be unlikely to infer, from the previous system output, the dataset from which an output is derived. This work introduces differential privacy to discrete event systems modeled by probabilistic automata to protect the state information pertaining to system resource configurations. State differential privacy is defined to protect the initial state of a discrete event system, which represents its initial resource configuration. Step-based state differential privacy verification is proposed in the framework of probabilistic automata, such that an attacker is unlikely to determine the initial state from which a system evolves, within a finite step of observations, if two systems with two different initial states satisfy state differential privacy. Specifically, the probability distributions of generating observations within a finite step from the two different initial states are approximate. If the two systems do not satisfy state differential privacy, a control specification is proposed, such that state differential privacy is enforced via supervisory control that is maximally permissive. Experimental studies are given to illustrate that the proposed method can effectively verify state differential privacy and enforce privacy protection in the probabilistic automata framework.

Keywords: differential privacy; discrete event system; probabilistic automaton; supervisory control; privacy protection

MSC: 93C65; 93E03

Citation: Teng, Y.; Li, Z.; Yin, L.; Wu, N. State-Based Differential Privacy Verification and Enforcement for Probabilistic Automata. *Mathematics* **2023**, *11*, 1853. https://doi.org/10.3390/math11081853

Academic Editor: Cheng-Chi Lee

Received: 10 March 2023
Revised: 11 April 2023
Accepted: 11 April 2023
Published: 13 April 2023

1. Introduction

With the extensive applications of computer communication and data mining technology, many organizations and institutes have studied the value of data by publishing user data sets, which contain personal private data. In the process of publishing the data sets, the privacy of users may be leaked, damaging the security of private information [1,2]. This requires publishers to anonymize or encrypt the data before publishing the security-sensitive data sets. Cryptography cannot only ensure the confidentiality of information, but also ensure its integrity and availability [3,4]. However, it reduces data processing speed and decreases data storage capacity. K-anonymity method anonymizes relational databases such that the individual information contained in the release cannot be distinguished from at least k-1 individuals whose information also appears in the release [5]. A (k, p)-anonymity extends k-anonymity to hide multiple pieces of sensitive information in transactions and sanitize transactional database with personalized sensitivity [6]. To reduce the computational complexity of k-anonymity, the authors in Ref. [7] proposed a novel efficient anonymization system to anonymize transactional data with a small information loss. These methods rely on the background knowledge of attackers. The more background knowledge attackers have, the higher the probability of privacy leakage is.

To solve this problem, the author in Ref. [8] proposed the notion of differential privacy, providing a mathematically strict method to protect sensitive personal information. Differential privacy protects sensitive data by adding random noise to the results to be inquired

by an attacker or a malicious observer [9–11]. To protect different data between two data sets that differ by one record, noise mechanisms are developed to achieve differential privacy by adding random noise satisfying different distributions to the query results, such as Laplace mechanism and Gauss mechanism [12–14]. These mechanisms are not suitable for protecting non-numerical sensitive data. The exponential mechanism is developed to achieve differential privacy for non-numerical data by randomly generating responses based on how well those responses approximate the non-private response [15]. An attacker or a malicious observer conducts a differential attack based on data sets that differ by one record, but the probabilities of two data sets outputting the same result are approximate.

Different from the private information protected by these privacy protection technologies, the sensitive information of *discrete event systems* (DESs) is the behavior information and resource configuration information, which is represented by the language and state, respectively. DESs refer to systems in which system states transit discretely at certain random points of time, thanks to the triggering of events [16]. An attacker or malicious observer can attack a DES by observing its behavior to infer the other sensitive information, such as the initial state. As an example, the initial state of an armored truck application system represents its initial location information, which should be kept confidential, otherwise it may enable potential hijackers to elaborate upon a perfect hijack plan.

The existing methods for protecting the sensitive information (e.g., the initial state and language) of a DES are developed through the formation of notion of state-based opacity and language-based opacity, where the secret is defined as a set of states and language, respectively [17,18]. For the language-based opacity, a system is said to be opaque with respect to a given secret if no execution leads to an estimate that is completely contained in the secret. Unlike language-based opaque verification, the authors in Ref. [19] developed an exponential mechanism that approximates a sensitive string (or word) using a randomly chosen string (or word) that is given by the Levenshtein distance, which is used to control the similarity or nearness of a sensitive string and its output counterpart. The work is extended to both real-time control and Markov chains for protecting trajectories generated by symbolic systems [20]. However, these methods do not consider the security of state privacy information of a symbolic control system. In this work, we focus on protecting the initial state of DESs modeled with probabilistic automata.

For the initial-state opacity (an important state-based security property), it is assumed that a malicious attacker (observer) fully knows the structure of a system, but only partially observes the event occurrences in it. Note that unobservable events are invisible to the attacker, and the attacker does not know the initial state unless it can be inferred by observations. Given a secret described by a set of states, a system is said to be initial-state opaque with respect to the secret if the attacker is never able to infer that the initial state of the system is within the secret [21–24]. The initial state information in the framework of probabilistic DESs cannot be protected by the existing methods of initial-state opacity if the occurrence likelihood of events is considered. With a metric on the states of a system, the authors in Ref. [25] formalized differential privacy by use of the probability ratio in the distributions after the same labeled transitions of relevant states. However, an attacker can infer the initial state based on the probability distribution of the language generated by the system via a long-term observation.

Automata and Petri nets are two typical tools to simulate the operation of DESs, where a finite state automaton is a machine that, given a symbolic input, transforms a series of states according to a transition function [26–28]. To the best of our knowledge, the verification and enforcement of differential privacy in DESs has not been well-defined and fully explored. The introduction of differential privacy into the framework of automata is of great significance for the protection of state private information, especially the protection of system initial states. To this end, differential privacy is introduced to the community of DESs that are, in this particular research, modeled by *probabilistic automata* [29,30].

This paper addresses the differential privacy problem in the framework of DESs modeled by probabilistic automata. The main contributions of this work can be summarized:

1. The notion of state differential privacy is formulated to protect the initial state information of a DES whose behaviors can be described by a probabilistic automaton. Two adjacent initial states are defined to represent the similar initial resource configurations. Step-based verification for state differential privacy is proposed to verify whether two probabilistic automata satisfy state differential privacy, within a finite step of observations, after the two systems generate a given observation from two adjacent initial states.
2. For two probabilistic automata with two adjacent initial states, a verifier is constructed to compute the probabilities of generating any same observation from the two adjacent initial states. If the two systems do not satisfy state differential privacy, we propose a control specification such that the state differential privacy is enforced to the closed-loop systems via supervisory control. The supervisory control is maximally permissive for enforcing privacy protection.
3. Through experimental studies, it is shown that the proposed method achieves state differential privacy in the considered class of automata and protects the initial state.

The rest of the paper is organized as follows. Section 2 introduces the backgrounds of probabilistic automata and the notion of differential privacy in sensitive data security. Section 3 is a problem statement. Step-based verification for state differential privacy is formulated in Section 4. Section 5 reports a method to enforce state differential privacy via supervisory control. A numerical example is shown in Section 6. Finally, Section 7 concludes this paper.

2. Preliminaries

In this section, we introduce probabilistic automata and the standard concept of differential privacy.

2.1. Probabilistic Automata

A *deterministic finite automaton* (DFA) is a four-tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0)$, where Q is a finite set of states, $\Sigma = \{\alpha, \beta, \cdots\} = \Sigma_o \dot{\cup} \Sigma_{uo}$ is an alphabet of finite events (Σ_o is the set of observable events and Σ_{uo} is that of unobservable events), $\delta : Q \times \Sigma \to Q$ is a partial transition function, and $q_0 \in Q$ is an initial state. The state transition function specifies the dynamics of the DFA: write $\delta(q, e)!$ if $e \in \Sigma$ can occur from state $q \in Q$, saying that $\delta(q, e)$ is defined.

The transition function δ is traditionally extended by induction on the length of strings to $\delta : Q \times \Sigma^* \to Q$ by defining $\delta(q, \varepsilon) = q$ and $\delta(q, se) = \delta(\delta(q, s), e)$ for $q \in Q$, $s \in \Sigma^*$ and $e \in \Sigma$, where $\delta(q, s)$ and $\delta(\delta(q, s), e)$ are both defined. Given a state $q \in Q$ and a string $s \in \Sigma^*$, write $\delta(q, s)!$ if δ is defined for s at q. The length of a string s, denoted by $|s|$, is the number of symbol occurrences in it.

The *generated language* of an automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0)$ from a state $q \in Q$ is defined as

$$\mathcal{L}(\mathcal{A}, q) = \{s \in \Sigma^* \mid \delta(q, s)!\}.$$

An attacker can only observe and record observable events. The natural projection $P : \Sigma^* \to \Sigma_o^*$ can be used to map any string executed in a system to the sequence of observable events, called an observation. This projection is defined recursively as $P(se) = P(s)P(e)$, $s \in \Sigma^*$, $e \in \Sigma$, with $P(e) = e$ if $e \in \Sigma_o$ and $P(e) = \varepsilon$ if $e \in \Sigma_{uo}$, where ε represents the empty string.

The *generated observations* of an automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0)$ from a state $q \in Q$ is defined as

$$\mathcal{L}_o(\mathcal{A}, q) = \{\omega \in \Sigma_o^* \mid s \in \mathcal{L}(\mathcal{A}, q) \ \& \ P(s) = \omega\}.$$

This paper explores the differential privacy problem on DESs modeled by probabilistic automata.

Definition 1 (Probabilistic automaton [29]). *A probabilistic automaton is a DFA equipped with a probability distribution of event occurrences, denoted by a two-tuple* $G = (\mathcal{A}, \rho)$*, where* $\mathcal{A} = (Q, \Sigma, \delta, q_0)$ *is a DFA and* $\rho : Q \times \Sigma \to [0, 1]$ *is a probability distribution function. Given a state* $q \in Q$ *and an event* $e \in \Sigma$*,* $\rho(q, e)$ *indicates the firing probability of* e *from* q *such that* $\rho(q, e) = 0$ *if* $\delta(q, e)$ *is not defined and* $\rho(q, e) > 0$ *if* $\delta(q, e)!$*. The set of all enabled (feasible) events at a state* $q \in Q$ *is denoted by* $\mathcal{E}(q) = \{e \in \Sigma \mid \rho(q, e) > 0\}$ *with* $\sum_{e \in \mathcal{E}(q)} \rho(q, e) = 1$*.*

In what follows, $G_s = (Q, \Sigma, \delta, \rho)$ is called a probabilistic automaton structure (or the skeleton of a probabilistic automaton), that is, a probabilistic automaton $G = (Q, \Sigma, \delta, q_0, \rho)$ is a probabilistic automaton structure $G_s = (Q, \Sigma, \delta, \rho)$ equipped with an initial state q_0. Write $G = (Q, \Sigma, \delta, q_0, \rho)$ as $G(q_0)$ if G_s is implicitly defined.

Example 1. *A probabilistic automaton structure* $G_s = (Q, \Sigma, \delta, \rho)$ *is shown in Figure 1. Given an initial state* q_1*,* $G(q_1)$ *is a probabilistic automaton with* $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$ *and* $\Sigma = \{\alpha, \beta, \lambda, \gamma, \mu, \tau\}$*, it holds* $\rho(q_1, \beta) = 0.4$ *and* $\rho(q_1, \lambda) = 0.6$*. Furthermore,* $\sum_{e \in \mathcal{E}(q_1)} \rho(q_1, e) = 1$ *with* $\mathcal{E}(q_1) = \{\beta, \lambda\}$*.*

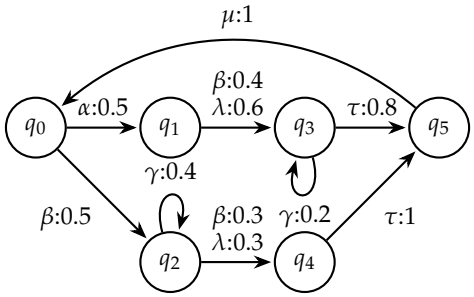


Figure 1. A probabilistic automaton structure G_s.

2.2. Differential Privacy

Generally speaking, a randomized algorithm is said to satisfy differential privacy if an attacker (or a malicious observer) is unlikely to distinguish between the outputs of two data sets differing on, at most, one element. Specifically, given a non-negative real number ϵ, a randomized algorithm $\mathcal{F}$ satisfies ϵ-differential privacy if, for any two input data sets D_1 and D_2 differing on, at most, one element (either $D_1 = D_2$ or there exists a datum d such that $D_1 \cup \{d\} = D_2$ or $D_2 \cup \{d\} = D_1$), and for any set of outputs O, it holds that [12]

$$e^{-\epsilon} \leq \frac{\mathbb{P}(\mathcal{F}(D_1) \in O)}{\mathbb{P}(\mathcal{F}(D_2) \in O)} \leq e^{\epsilon}.$$

Note that $\mathcal{F}(D_1)$ (or $\mathcal{F}(D_2)$) is the output of $\mathcal{F}$ on input D_1 (or D_2), $\mathbb{P} : O \to (0, 1]$ is the probability function, mapping an output of $\mathcal{F}$ to a real number between zero and one (including one), and ϵ is the privacy budget parameter that stipulates the level of privacy protection with $\epsilon \in \mathbb{R}$ and $\epsilon \geq 0$, where $\mathbb{R}$ is the set of real numbers.

3. Problem Statement

This section introduces differential privacy into the framework of probabilistic automata to protect the initial state information. We first define state differential privacy and establish its mathematical developments below.

3.1. State Differential Privacy

Given a probabilistic automaton $G = (Q, \Sigma, \delta, q_0, \rho)$, the probability of generating string $se \in \Sigma^*$ from state q with $s \in \Sigma^*$ and $e \in \Sigma$ is recursively defined as

$$\Pr_\sigma(q, se) = \begin{cases} 1, \text{ if } se = \varepsilon \\ \Pr_\sigma(q, s) \times \rho(\delta(q, s), e), \text{ if } \delta(q, s)! \\ 0, \text{ otherwise} \end{cases}$$

where $\varepsilon \in \Sigma^*$ is the empty string. Intuitively, $\Pr_\sigma(q, s)$ could be viewed as the probability that the string s can be executed from q in plant G. $\Pr_\sigma(q, s) > 0$ if $\delta(q, s)!$ for $s \in \Sigma^*$.

The *generated language* of a probabilistic automaton $G = (Q, \Sigma, \delta, q_0, \rho)$ from a state $q \in Q$ is defined as

$$\mathcal{L}(G, q) = \{s \in \Sigma^* \mid \Pr_\sigma(q, s) > 0\}.$$

Accordingly, the *generated observations* of a probabilistic automaton $G = (Q, \Sigma, \delta, q_0, \rho)$ from a state $q \in Q$ is defined as

$$\mathcal{L}_o(G, q) = \{\omega \in \Sigma_o^* \mid \exists s \in \mathcal{L}(G, q) : \ \omega = P(s)\}.$$

The set of strings that are consistent with observation $\omega \in \Sigma_o^*$ generated at state q is defined as

$$S(q, \omega) = \{s \in \Sigma^* \mid s \in \mathcal{L}(G, q) \ \& \ P(s) = \omega\}.$$

The probability of generating observation $\omega \in \Sigma_o^*$ from a state q is

$$\Pr_o(q, \omega) = \sum_{s \in S(q, \omega)} \Pr_\sigma(q, s).$$

Example 2. *Consider the probabilistic automaton structure in Figure 1 with initial state q_0. Suppose $\Sigma_o = \{\alpha, \beta, \lambda, \gamma, \mu\}$ and $\Sigma_{uo} = \{\tau\}$. Given a string $s = \beta\gamma\beta\tau$, the probability of generating s from q_0 is denoted as $\Pr_\sigma(q_0, \beta\gamma\beta\tau) = \rho(q_0, \beta) \times \rho(q_2, \gamma) \times \rho(q_2, \beta) \times \rho(q_4, \tau) = 0.06$. Let $\omega = \alpha\beta\gamma$. $S(q_0, \omega) = \{\alpha\beta\gamma, \alpha\beta\gamma\tau\}$ and $\Pr_o(q_0, \omega) = \Pr_\sigma(q_0, \alpha\beta\gamma) + \Pr_\sigma(q_0, \alpha\beta\gamma\tau) = 0.072$ hold.*

Given a probabilistic automaton $G = (Q, \Sigma, \delta, q_0, \rho)$, the set of states reached by generating any string that is consistent with observation $\omega \in \Sigma_o^*$ from a state q is denoted by

$$\varphi(q, \omega) = \{q' \in Q \mid \exists s \in S(q, \omega) : \ \delta(q, s) = q'\}.$$

Let $q \in \varphi(q_0, \omega)$ be a reachable state after the system generates an observation $\omega \in \Sigma_o^*$ from initial state q_0. The probability of generating $s \in \Sigma^*$ from q is defined as

$$\Pr_\sigma(q_0, \omega, q, s) = \Pr(q|\varphi(q_0, \omega)) \times \Pr_\sigma(q, s)$$

where $\Pr(q|\varphi(q_0, \omega))$ is the probability of choosing q from $\varphi(q_0, \omega)$, defined as

$$\Pr(q|\varphi(q_0, \omega)) = \begin{cases} \dfrac{\sum\limits_{s \in S(q_0, \omega) \& \delta(q_0, s) = q} \Pr_\sigma(q_0, s)}{\sum\limits_{s \in S(q_0, \omega)} \Pr_\sigma(q_0, s)}, q \in \varphi(q_0, \omega) \\ 0, \ q \notin \varphi(q_0, \omega) \end{cases}$$

In conclusion, $\sum_{q \in Q} \Pr(q|\varphi(q_0, \omega)) = 1$ holds.

The probability of choosing q from $\varphi(q_0, \omega)$, denoted by $\Pr(q|\varphi(q_0, \omega))$, is the ratio of the sum of probabilities of the strings that are, consistent with an observation ω, generated from q_0, reaching state q, with the sum of probabilities of the strings that are, consistent with ω, generated from q_0. In brief, $\Pr(q|\varphi(q_0, \omega))$ is the probability that the system reaches state q (from q_0) under the premise of generating an observation ω.

Example 3. *Consider the probabilistic automaton structure in Figure 1 with initial state q_1. Suppose that $\Sigma_o = \{\alpha, \beta, \lambda, \gamma, \mu\}$ and $\Sigma_{uo} = \{\tau\}$. We have $\varphi(q_1, \beta) = \{q_3, q_5\}$ and $S(q_1, \beta) = \{\beta, \beta\tau\}$. For $s \in S(q_1, \beta)$ and $\delta(q_1, s) = q_3$, it holds $\sum_s \Pr_\sigma(q_1, s) = \rho(q_1, \beta) = 0.4$. For $s \in S(q_1, \beta)$ and $\delta(q_1, s) = q_5$, it holds that $\sum_s \Pr_\sigma(q_1, s) = \rho(q_1, \beta) \times \rho(q_3, \tau) = 0.32$. The probabilities of choosing states q_3 and q_5 from $\varphi(q_1, \beta)$ are $\Pr(q_3|\varphi(q_1, \beta)) = 0.4/(0.4 + 0.32) = 5/9$ and*

$\Pr(q_5|\varphi(q_1,\beta)) = 0.32/(0.4+0.32) = 4/9$, *respectively. We obtain* $\Pr_\sigma(q_1,\beta,q_3,\tau\mu\alpha) = 5/9\times 0.8\times 1\times 0.5 = 2/9$ *and* $\Pr_\sigma(q_1,\beta,q_5,\mu\alpha\beta) = 4/9\times 1\times 0.5\times 0.4 = 4/45$.

Let $\mathbb{N}$ be the set of natural numbers and $\mathbb{N}^+ = \{x>0 | x\in\mathbb{N}\}$. Given an observation $\omega\in\Sigma_o^*$, the set of all observations generated from a state $q\in\varphi(q_0,\omega)$ with $k\in\mathbb{N}^+$ steps is defined as

$$\mathcal{L}_o(q_0,\omega,q,k) = \{\omega'\in\Sigma_o^* \mid \omega'\in\mathcal{L}_o(G,q)\ \&\ |\omega'| = k\}.$$

The set of all observations due to $k\in\mathbb{N}^+$-step observation extensions in a system after generating an observation ω from q_0, is defined as

$$\mathcal{L}_o(q_0,\omega,k) = \bigcup_{q\in\varphi(q_0,\omega)} \mathcal{L}_o(q_0,\omega,q,k).$$

The probability of generating $\omega'\in\mathcal{L}_o(q_0,\omega,k)$ after the system generates an observation ω from q_0 is

$$\Pr_o(q_0,\omega,k,\omega') = \sum_{q\in\varphi(q_0,\omega)}\ \sum_{s\in S(q,\omega')} \Pr_\sigma(q_0,\omega,q,s).$$

Note that $\Pr_o(q_0,\omega,k,\omega')$ is the probability of generating an observation ω' due to a k-step observation extension, under the premise that an observation ω from q_0 has been generated.

Example 4. *Let us consider the probabilistic automaton structure in Figure 1 with initial state* q_1. *Suppose* $\Sigma_o = \{\alpha,\beta,\lambda,\gamma,\mu\}$, $\Sigma_{uo} = \{\tau\}$ *and* $k=2$. *We have* $\mathcal{L}_o(q_1,\beta,q_3,2) = \{\gamma\gamma,\ \gamma\mu,\ \mu\alpha,\ \mu\beta\}$ *and* $\mathcal{L}_o(q_1,\beta,q_5,2) = \{\mu\alpha,\ \mu\beta\}$. *Due to* $\mathcal{L}_o(q_1,\beta,2) = \mathcal{L}_o(q_1,\beta,q_3,2)\cup\mathcal{L}_o(q_1,\beta,q_5,2) = \{\gamma\gamma,\ \gamma\mu,\ \mu\alpha,\ \mu\beta\}$, *the probabilities of generating* $\omega'\in\mathcal{L}_o(q_1,\beta,2)$ *after the system generates an observation* $\omega=\beta$ *from state* q_1 *is*

$$\begin{aligned}
&\textit{For } \omega' = \gamma\gamma:\ \Pr_o(q_1,\beta,2,\gamma\gamma) = \Pr_\sigma(q_1,\beta,q_3,\gamma\gamma) = 5/9\times 0.2\times 0.2 = 1/45;\\
&\textit{For } \omega' = \gamma\mu:\ \Pr_o(q_1,\beta,2,\gamma\mu) = \Pr_\sigma(q_1,\beta,q_3,\gamma\tau\mu) = 5/9\times 0.2\times 0.8\times 1 = 4/45;\\
&\textit{For } \omega' = \mu\alpha:\ \Pr_o(q_1,\beta,2,\mu\alpha) = \Pr_\sigma(q_1,\beta,q_3,\tau\mu\alpha) + \Pr_\sigma(q_1,\beta,q_5,\mu\alpha)\\
&\qquad\qquad = 5/9\times 0.8\times 1\times 0.5 + 4/9\times 1\times 0.5 = 4/9;\\
&\textit{For } \omega' = \mu\beta:\ \Pr_o(q_1,\beta,2,\mu\beta) = \Pr_\sigma(q_1,\beta,q_3,\tau\mu\beta) + \Pr_\sigma(q_1,\beta,q_5,\mu\beta)\\
&\qquad\qquad = 5/9\times 0.8\times 1\times 0.5 + 4/9\times 1\times 0.5 = 4/9.
\end{aligned}$$

Definition 2 (Adjacent states)**.** *Given a probabilistic automaton structure* $G_s = (Q,\Sigma,\delta,\rho)$ *and two initial states* $q_0\in Q$ *and* $q_0'\in Q$, q_0 *and* q_0' *are said to be adjacent if there exists an observation* $\omega\in\Sigma_o^*\setminus\{\varepsilon\}$ *such that* $\Pr_o(q_0,\omega)>0$ *and* $\Pr_o(q_0',\omega)>0$ *hold.*

Two initial states q_0 and q_0' are adjacent if an observation that is not the empty string can be generated from both q_0 and q_0'. The concept of state differential privacy in probabilistic automata is presented to conceal two adjacent initial states of a system.

Definition 3 (State differential privacy)**.** *Given a probabilistic automaton structure* $G_s = (Q,\Sigma,\delta,\rho)$ *and two adjacent initial states* q_0, q_0' *(leading to two probabilistic automata* $G(q_0)$ *and* $G(q_0')$*),* $G(q_0)$ *and* $G(q_0')$ *are said to satisfy* ϵ*-state differential privacy, within a given* k*-step observation extension if, after* $G(q_0)$ *and* $G(q_0')$ *generate an observation* $\omega\in\Sigma_o^*$ *from* q_0 *and* q_0'*, respectively, for all* $k'\le k$, *for all* $\omega'\in\mathcal{L}_o(q_0,\omega,k')\cup\mathcal{L}_o(q_0',\omega,k')$, *it holds*

$$|\Pr_o(q_0,\omega,k',\omega') - \Pr_o(q_0',\omega,k',\omega')| \le \epsilon,$$

where the parameter ϵ is a positive real number between zero and one, which stipulates the level of privacy protection for adjacent initial states.

3.2. Problems

In this subsection, we formulate two problems involving the pre-defined state differential privacy in probabilistic automata. First, this work focuses on step-based verification for state differential privacy to protect the initial state.

Problem 1. *Given two probabilistic automata $G(q_0) = (Q, \Sigma, \delta, q_0, \rho)$ and $G(q_0') = (Q, \Sigma, \delta, q_0', \rho)$, and an observation $\omega \in \Sigma_o^*$, construct a verifier $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$ that is a finite state automaton to verify whether $G(q_0)$ and $G(q_0')$ satisfy ϵ-state differential privacy, within k-step observation extensions, after $G(q_0)$ and $G(q_0')$ generate the given observation ω from two adjacent initial states q_0 and q_0', respectively.*

Next, we propose a supervisory control method to supervise the behavior of two probabilistic automata such that the controlled systems satisfy state differential privacy.

Problem 2. *Given a verifier $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$ for two probabilistic automata $G(q_0)$ and $G(q_0')$, find a supervisor such that the controlled systems satisfy ϵ-state differential privacy while the supervisory control is maximally permissive.*

A solution to Problem 2 would ensure that an attacker is unlikely to infer the initial states of the two probabilistic automata within a given k-step observation extension.

4. Step-Based Verification for State Differential Privacy

This section provides a step-based verification method for state differential privacy in DESs modeled with probabilistic automata. The information of similar initial resource configurations of two probabilistic automata is protected if the two systems satisfy state differential privacy, within a finite step observation extension, after the two systems generate a given observation from two given adjacent initial states.

The set of post-states of a state q in a probabilistic automaton $G = (Q, \Sigma, \delta, q_0, \rho)$ is defined as

$$q^\sharp = \{q' \in Q \mid (\exists e \in \Sigma)\ \delta(q, e) = q'\}.$$

Given a probabilistic automaton $G = (Q, \Sigma, \delta, q_0, \rho)$, for a state $q \in Q$ and an observable event $e \in \Sigma_o$, we define

$$\sigma(q, e) = \{s \in \Sigma^* \mid (\exists t \in \Sigma_{uo}^*) s = te, \Pr_\sigma(q, s) > 0\}$$

as the set of *extended strings* of e generated at q.

All extended strings of an observable event generated at a state are the concatenation of an unobservable string or the empty string with the observable event, and must end with the observable event. Given a positive integer k, suppose that an observation ω has been generated from two adjacent initial states. When a new observation ω' occurs, whose length is less than or equal to k, $\omega\omega'$ is observed. A verifier needs to be defined to check whether two systems with two adjacent initial states satisfy state differential privacy.

Definition 4 (Verifier)**.** *Given two probabilistic automata $G(q_0) = (Q, \Sigma, \delta, q_0, \rho)$ and $G(q_0') = (Q, \Sigma, \delta, q_0', \rho)$, and an observation $\omega \in \Sigma_o^*$, a verifier is a four-tuple $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$, where Q_v is a finite set of all states $\mathcal{S}_1 \times \mathcal{S}_2$ with $\mathcal{S}_1, \mathcal{S}_2 \in 2^Q$, $\mathcal{Q}_0 = \varphi(q_0, \omega) \times \varphi(q_0', \omega)$ is an initial state, Σ_o is a set of observable events, and $\delta_v : Q_v \times \Sigma_o \to Q_v$ is a state transition function such that, $\forall \mathcal{S}_1 \in 2^Q$, $\forall \mathcal{S}_2 \in 2^Q$ with $\mathcal{S}_1 \neq \mathcal{S}_2$, $\forall e \in \Sigma_o$, for $\mathcal{Q} = \mathcal{S}_1 \times \mathcal{S}_2 \in Q_v$ with $\mathcal{Q} \neq \emptyset$, $\bigcup_{q \in \mathcal{S}_1 \cup \mathcal{S}_2} \varphi(q, e) \neq \emptyset \Rightarrow \bigcup_{q \in \mathcal{S}_1} \varphi(q, e) \times \bigcup_{q \in \mathcal{S}_2} \varphi(q, e) = \delta_v(\mathcal{Q}, e) \in Q_v$. Write $\delta_v(\mathcal{Q}, e)!$ if δ_v is defined for an observable event $e \in \Sigma_o$ at state $\mathcal{Q} \in Q_v$.*

Given two probabilistic automata $G(q_0) = (Q, \Sigma, \delta, q_0, \rho)$ and $G(q_0') = (Q, \Sigma, \delta, q_0', \rho)$, and an observation ω, a verifier $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$ is constructed by Algorithm 1. For two adjacent initial states q_0 and q_0', lines 1–15 compute $\varphi(q_0, \omega)$, $\varphi(q_0', \omega)$ and take their Cartesian product as an initial state $\mathcal{Q}_0$ (Q_v is initialized to $\{\mathcal{Q}_0\}$). For $\mathcal{Q} = \mathcal{S}_1 \times \mathcal{S}_2 \in Q_n$ (Q_n is initialized to $\{\mathcal{Q}_0\}$), if $\mathcal{S}_1 \neq \mathcal{S}_2$ and $\mathcal{Q} \neq \emptyset$, for any observable event $e \in \Sigma_o$, we obtain the sets of all reachable states by generating e from all states in $\mathcal{S}_1$ and $\mathcal{S}_2$, denoted by $\mathcal{S}_1'$ and $\mathcal{S}_2'$, respectively. $\delta_v(\mathcal{Q}, e) = \mathcal{S}_1' \times \mathcal{S}_2'$ is defined, $\mathcal{S}_1' \times \mathcal{S}_2'$ is inserted into Q_n and Q_v, and $\mathcal{Q}$ is removed from Q_n. Recursively execute what is stated as above until Q_n is the empty set. The complexity of Algorithm 1 is $\mathcal{O}(2^{|Q|})$.

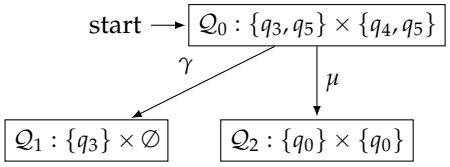
Algorithm 1: Construction of a verifier

Input: Two probabilistic automata $G(q_0) = (Q, \Sigma, \delta, q_0, \rho)$ and $G(q_0') = (Q, \Sigma, \delta, q_0', \rho)$, and an observation ω

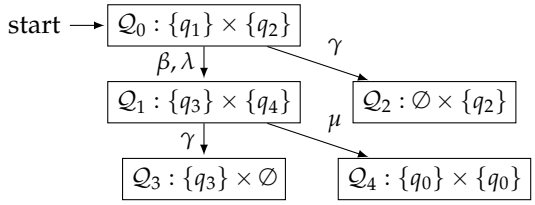
Output: A verifier $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$

1 **foreach** $q \in \{q_0, q_0'\}$ **do**
2 $\quad e \leftarrow \omega[1]$; $\varphi(q, e) \leftarrow \emptyset$; $S_q \leftarrow \{q\}$;
3 $\quad$ **foreach** $q' \in S_q$ **do**
4 $\quad\quad S_q \leftarrow S_q \setminus \{q'\}$;
5 $\quad\quad$ **foreach** $e' \in \{e\} \cup \Sigma_{uo}$ **do**
6 $\quad\quad\quad$ **if** $\delta(q', e') = q''$ & $e' = e$ **then**
7 $\quad\quad\quad\quad \varphi(q, e) \leftarrow \varphi(q, e) \cup \{q''\}$;
8 $\quad\quad\quad$ **if** $\delta(q', e') = q''$ & $e' \in \Sigma_{uo}$ **then**
9 $\quad\quad\quad\quad S_q \leftarrow S_q \cup \{q''\}$;
10 $\quad$ **for** $i = 2$, $i \leq |\omega|$; $i++$ **do**
11 $\quad\quad \omega' \leftarrow \omega[1] \cdots \omega[i-1]$; $\omega'' \leftarrow \omega'\omega[i]$; $\varphi(q, \omega'') \leftarrow \emptyset$;
12 $\quad\quad$ **foreach** $q' \in \varphi(q, \omega')$ **do**
13 $\quad\quad\quad e \leftarrow \omega[i]$; $\varphi(q', e) \leftarrow \emptyset$; $S_q \leftarrow \{q'\}$;
14 $\quad\quad\quad$ compute $\varphi(q', \omega[i])$ by lines 3–9;
15 $\quad\quad\quad \varphi(q, \omega'') \leftarrow \varphi(q, \omega'') \cup \varphi(q', \omega[i])$;
16 $\mathcal{Q}_0 \leftarrow \varphi(q_0, \omega) \times \varphi(q_0', \omega)$; $Q_v \leftarrow \{\mathcal{Q}_0\}$; $Q_n \leftarrow \{\mathcal{Q}_0\}$;
17 **foreach** $\mathcal{Q} = \mathcal{S}_1 \times \mathcal{S}_2 \in Q_n$ **do**
18 $\quad$ **if** $\mathcal{S}_1 \neq \mathcal{S}_2$ & $\mathcal{Q} \neq \emptyset$ **then**
19 $\quad\quad$ **foreach** $e \in \Sigma_o$ **do**
20 $\quad\quad\quad$ **foreach** $q \in \mathcal{S}_1 \cup \mathcal{S}_2$ **do**
21 $\quad\quad\quad\quad \varphi(q, e) \leftarrow \emptyset$; $S_q \leftarrow \{q\}$;
22 $\quad\quad\quad\quad$ compute $\varphi(q, e)$ by lines 3–9;
23 $\quad\quad\quad \mathcal{S}_1' \leftarrow \bigcup_{q \in \mathcal{S}_1} \varphi(q, e)$; $\mathcal{S}_2' \leftarrow \bigcup_{q \in \mathcal{S}_2} \varphi(q, e)$;
24 $\quad\quad\quad \mathcal{Q}' \leftarrow \mathcal{S}_1' \times \mathcal{S}_2'$; $\mathcal{Q}' \leftarrow \delta_v(\mathcal{Q}, e)$;
25 $\quad\quad\quad Q_v \leftarrow Q_v \cup \{\mathcal{Q}'\}$; $Q_n \leftarrow Q_n \cup \{\mathcal{Q}'\}$;
26 $\quad Q_n \leftarrow Q_n \setminus \{\mathcal{Q}\}$;

Example 5. *Let us consider the probabilistic automaton structure in Figure 1 with two adjacent initial states q_1 and q_2. Suppose $\Sigma_o = \{\alpha, \beta, \lambda, \gamma, \mu\}$ and $\Sigma_{uo} = \{\tau\}$. If two systems $G(q_1)$ and $G(q_2)$ generate an observation β, $\varphi(q_1, \beta) = \{q_3, q_5\}$ and $\varphi(q_2, \beta) = \{q_4, q_5\}$ hold. $\mathcal{Q}_0 = \{q_3, q_5\} \times \{q_4, q_5\}$ is an initial state in a verifier V_β. For state $\mathcal{Q}_0$ and observable event γ, $\bigcup_{q \in \{q_3, q_5\}} \varphi(q, \gamma) = \{q_3\}$, $\bigcup_{q' \in \{q_4, q_5\}} \varphi(q', \gamma) = \emptyset$ and $\delta_v(\mathcal{Q}_0, \gamma) = \{q_3\} \times \emptyset$ hold. For state $\mathcal{Q}_0$ and observable event μ, it holds $\bigcup_{q \in \{q_3, q_5\}} \varphi(q, \mu) = \{q_0\}$, $\bigcup_{q' \in \{q_4, q_5\}} \varphi(q', \mu) = \{q_0\}$ and $\delta_v(\mathcal{Q}_0, \mu) = \{q_0\} \times \{q_0\}$. A verifier V_β is shown in Figure 2a. If $G(q_1)$ and $G(q_2)$ do not generate any observation before verifying state differential privacy, a verifier V_ε is shown in Figure 2b.*



(a) Verifier V_β.



(b) Verifier V_ε

Figure 2. Verifiers V_β and V_ε for $G(q_1)$ and $G(q_2)$.

Given a verifier $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$, a sequence of states and observable events $\mathcal{Q}_{i0} e_{i1} \mathcal{Q}_{i1} \cdots \mathcal{Q}_{ij-1} e_{ij} \mathcal{Q}_{ij}$ generates an observation $e_{i1} \cdots e_{ij}$ for $h = \{0, 1, \ldots, j\}$ and $e_{ih} \in \Sigma_o$, $\delta_v(\mathcal{Q}_{ih-1}, e_{ih}) = \mathcal{Q}_{ih}$ for $h = \{1, 2, \ldots, j\}$. The state transition function δ_v is extended to $\delta_v : Q_v \times \Sigma_o^* \to Q_v$ by defining $\delta_v(\mathcal{Q}, \varepsilon) = \mathcal{Q}$ and $\delta_v(\mathcal{Q}, se) = \delta_v(\delta_v(\mathcal{Q}, s), e)$ for $\mathcal{Q} \in Q_v$, $s \in \Sigma_o^*$ and $e \in \Sigma_o$.

For state $\mathcal{Q} \in Q_v$, ${}^\sharp\mathcal{Q}$ and $\mathcal{Q}^\sharp$ are the sets of pre- and post-states of $\mathcal{Q}$, respectively, defined as

$$
\begin{aligned}
{}^\sharp\mathcal{Q} &= \{\mathcal{Q}' \in Q_v \mid (\exists e \in \Sigma_o)\ \delta_v(\mathcal{Q}', e) = \mathcal{Q}\};\\
\mathcal{Q}^\sharp &= \{\mathcal{Q}' \in Q_v \mid (\exists e \in \Sigma_o)\ \delta_v(\mathcal{Q}, e) = \mathcal{Q}'\}.
\end{aligned}
$$

Proposition 1. *Given a probabilistic automaton structure $G_s = (Q, \Sigma, \delta, \rho)$, two adjacent initial states q_0 and q_0', and an observation $\omega \in \Sigma_o^*$, let $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$ be the verifier due to Algorithm 1. Given $\mathcal{S}_1 \in 2^Q$, $\mathcal{S}_2 \in 2^Q$, and a positive real number ϵ, for $\mathcal{Q} = \mathcal{S}_1 \times \mathcal{S}_2 \in Q_v$ with $\mathcal{Q}^\sharp = \emptyset$, if $|\mathrm{Pr}_o(q_0, \omega\omega') - \mathrm{Pr}_o(q_0', \omega\omega')| \le \epsilon$ where $\delta_v(\mathcal{Q}_0, \omega') = \mathcal{Q}$ and $\omega' \in \Sigma_o^*$, then $|\mathrm{Pr}_o(q_0, \omega\omega'\omega'') - \mathrm{Pr}_o(q_0', \omega\omega'\omega'')| \le \epsilon$ where $\forall \omega'' \in \Sigma_o^*$.*

Proof. For $\mathcal{Q} = \mathcal{S}_1 \times \mathcal{S}_2 \in Q_v$ in verifier V_ω, if $\mathcal{Q}^\sharp = \emptyset$, it holds that $\mathcal{S}_1 = \emptyset$, $\mathcal{S}_2 = \emptyset$ or $\mathcal{S}_1 = \mathcal{S}_2$. If $\mathcal{S}_1 = \emptyset$ and $|\mathrm{Pr}_o(q_0, \omega\omega') - \mathrm{Pr}_o(q_0', \omega\omega')| \le \epsilon$, then $\mathrm{Pr}_o(q_0, \omega\omega') = 0$ and $\mathrm{Pr}_o(q_0', \omega\omega') \le \epsilon$ hold. For any observation $\omega'' \in \Sigma_o^*$ generated from $q \in \mathcal{S}_2$, $0 \le \mathrm{Pr}_o(q, \omega'') \le 1$ and $\mathrm{Pr}_o(q_0, \omega\omega'\omega'') = 0$ hold. We have $\mathrm{Pr}_o(q_0', \omega\omega') \times \mathrm{Pr}_o(q, \omega'') \le \epsilon$, that is, $\mathrm{Pr}_o(q_0', \omega\omega'\omega'') \le \epsilon$. $|\mathrm{Pr}_o(q_0, \omega\omega'\omega'') - \mathrm{Pr}_o(q_0', \omega\omega'\omega'')| \le \epsilon$ holds. If $\mathcal{S}_2 = \emptyset$, we have $\mathrm{Pr}_o(q_0, \omega\omega'\omega'') \le \epsilon$, $\mathrm{Pr}_o(q_0', \omega\omega'\omega'') = 0$ and $|\mathrm{Pr}_o(q_0, \omega\omega'\omega'') - \mathrm{Pr}_o(q_0', \omega\omega'\omega'')| \le \epsilon$ for any observation $\omega'' \in \Sigma_o^*$ by the similar way. If $\mathcal{S}_1 = \mathcal{S}_2$ and $|\mathrm{Pr}_o(q_0, \omega\omega') - \mathrm{Pr}_o(q_0', \omega\omega')| \le \epsilon$, for any observation $\omega'' \in \Sigma_o^*$ generated from $q \in \mathcal{S}_1$ (or $q \in \mathcal{S}_2$), since $0 \le \mathrm{Pr}_o(q, \omega'') \le 1$, it holds that $|\mathrm{Pr}_o(q_0, \omega\omega') \times \mathrm{Pr}_o(q, \omega'') - \mathrm{Pr}_o(q_0', \omega\omega') \times \mathrm{Pr}_o(q, \omega'')| \le \epsilon$, that is, $|\mathrm{Pr}_o(q_0, \omega\omega'\omega'') - \mathrm{Pr}_o(q_0', \omega\omega'\omega'')| \le \epsilon$. □

In Proposition 1, for all states $\mathcal{Q} \in Q_v$ without post-states in verifier V_ω, given a positive real number ϵ, if the difference between the probabilities of generating $\omega\omega'$ from q_0 and q_0' is less than or equal to ϵ, where $\delta_v(\mathcal{Q}_0, \omega') = \mathcal{Q}$, then the difference between the probabilities of generating observation $\omega\omega'\omega''$ for all $\omega'' \in \Sigma_o^*$ from q_0 and q_0' is less than or equal to ϵ. The two systems with adjacent initial states verified by the verifier satisfy state differential privacy within any finite step observation extension.

Step-based verification for state differential privacy is implemented by Algorithm 2. Given a verifier $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$ for $G(q_0) = (Q, \Sigma, \delta, q_0, \rho)$ and $G(q_0') = (Q, \Sigma, \delta, q_0', \rho)$, a finite step $k \in \mathbb{N}^+$ and a parameter ϵ, for any $k' \le k$ with $k' \in \mathbb{N}^+$, for a state $\mathcal{Q} \in Q_n$ (Q_n is initialized to $\{\mathcal{Q}_0\}$), and for any observable event e with $\delta_v(\mathcal{Q}, e)!$, we obtain the

probabilities of generating $\omega' e$ with $\delta_v(\mathcal{Q}_0, \omega') = \mathcal{Q}$ after the systems generate ω from q_0 and q_0', respectively, denoted by $\Pr_1(\mathcal{Q}_0, \omega' e)$ and $\Pr_2(\mathcal{Q}_0, \omega' e)$. If the difference between $\Pr_1(\mathcal{Q}_0, \omega' e)$ and $\Pr_2(\mathcal{Q}_0, \omega' e)$ is larger than ϵ, the two systems with adjacent initial states do not satisfy ϵ-state differential privacy; otherwise, $\mathcal{Q}$ is deleted from Q_n and all post-states of $\mathcal{Q}$ are inserted into Q_n. Its complexity is $\mathcal{O}(k \times |Q_v|^2 \times |\Sigma_o| \times 2^{|Q|})$.

Algorithm 2: Step-based verification for state differential privacy

Input: A verifier $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$, a finite step k and a positive parameter ϵ
Output: True or False

$Q_n \leftarrow \{\mathcal{Q}_0\}$; $\Pr_1(\mathcal{Q}_0, \varepsilon) \leftarrow 1$; $\Pr_2(\mathcal{Q}_0, \varepsilon) \leftarrow 1$;
for $k' = 1$, $k' \leq k$; $k'++$ **do**
  foreach $\mathcal{Q} = \mathcal{S}_1 \times \mathcal{S}_2 \in Q_n$ & $\delta_v(\mathcal{Q}_0, \omega') = \mathcal{Q}$ **do**
    obtain $\varphi(q_0, \omega\omega')$, $\varphi(q_0', \omega\omega')$ by Algorithm 1;
    foreach $e \in \Sigma_o$ & $\delta_v(\mathcal{Q}, e)!$ **do**
      foreach $q \in \mathcal{S}_1 \cup \mathcal{S}_2$ **do**
        $\sigma(q, e) \leftarrow \emptyset$; $\omega_a \leftarrow \varepsilon$; $S_q \leftarrow \{(\omega_a, q)\}$;
        foreach $(\omega_a, q_a) \in S_q$ **do**
          $S_q \leftarrow S_q \setminus \{(\omega_a, q_a)\}$;
          foreach $e' \in \{e\} \cup \Sigma_{uo}$ **do**
            $\omega_a \leftarrow \omega_a e'$;
            if $\delta(q_a, e') = q_b$ & $e' = e$ **then**
              $\sigma(q, e) \leftarrow \sigma(q, e) \cup \{\omega_a\}$;
            if $\delta(q_a, e') = q_b$ & $e' \in \Sigma_{uo}$ **then**
              $S_q \leftarrow S_q \cup \{(\omega_a, q_b)\}$;
      $P_1 \leftarrow 0$; $P_2 \leftarrow 0$; $q_1 \leftarrow q_0$; $q_2 \leftarrow q_0'$;
      foreach $q \in \mathcal{S}_i$ & $s \in \sigma(q, e)$ & $i \in \{1, 2\}$ **do**
        $P_i \leftarrow P_i + \Pr(q|\varphi(q_i, \omega\omega')) \times \Pr_\sigma(q, s)$;
      foreach $i \in \{1, 2\}$ **do**
        $\Pr_i(\mathcal{Q}_0, \omega' e) \leftarrow \Pr_i(\mathcal{Q}_0, \omega') \times P_i$;
      if $|\Pr_1(\mathcal{Q}_0, \omega' e) - \Pr_2(\mathcal{Q}_0, \omega' e)| > \epsilon$ **then**
        return false;
    $\mathcal{Q}^\sharp \leftarrow \emptyset$;
    foreach $e \in \Sigma_o$ **do**
      $\mathcal{Q}^\sharp \leftarrow \mathcal{Q}^\sharp \cup \{\delta_v(\mathcal{Q}, e)\}$;
    $Q_n \leftarrow (Q_n \setminus \{\mathcal{Q}\}) \cup \mathcal{Q}^\sharp$;
return true;

Example 6. *Let us consider the verifier V_β in Figure 2a. Suppose $k = 3$ and $\epsilon = 0.12$. After the two probabilistic automata $G(q_1)$ and $G(q_2)$ generate an observation β from two adjacent initial states q_1 and q_2, $\varphi(q_1, \beta) = \{q_3, q_5\}$ and $\varphi(q_2, \beta) = \{q_4, q_5\}$ hold. The probabilities of choosing states q_3 and q_5 from $\varphi(q_1, \beta)$ are $\Pr(q_3|\varphi(q_1, \beta)) = 5/9$ and $\Pr(q_5|\varphi(q_1, \beta)) = 4/9$, respectively. The probabilities of choosing states q_4 and q_5 from $\varphi(q_2, \beta)$ are $\Pr(q_4|\varphi(q_2, \beta)) = 1/2$ and $\Pr(q_5|\varphi(q_2, \beta)) = 1/2$, respectively.*

For initial state $\mathcal{Q}_0$ and $k' = 1$, it holds that

$$|\Pr_1(\mathcal{Q}_0, \gamma) - \Pr_2(\mathcal{Q}_0, \gamma)| = \Pr(q_3|\{q_3, q_5\}) \times \Pr_\sigma(q_3, \gamma) = 1/9 \leq \epsilon;$$
$$\begin{aligned}|\Pr_1(\mathcal{Q}_0, \mu) - \Pr_2(\mathcal{Q}_0, \mu)| = |&\Pr(q_5|\{q_3, q_5\}) \times \Pr_\sigma(q_5, \mu) + \Pr(q_3|\{q_3, q_5\}) \times \Pr_\sigma(q_3, \tau\mu) \\ &- \Pr(q_5|\{q_4, q_5\}) \times \Pr_\sigma(q_5, \mu) - \Pr(q_4|\{q_4, q_5\}) \times \Pr_\sigma(q_4, \tau\mu)| = 1/9 \leq \epsilon.\end{aligned}$$

For $k' = 2$ and $k' = 3$, there is no state transition with k'-step observation extensions in V_β. Two systems $G(q_1)$ and $G(q_2)$ satisfy ϵ-state differential privacy with $\epsilon = 0.12$, within three-step observation extensions, after the systems generate β from q_1 and q_2.

Consider the verifier V_ε in Figure 2b. For initial state $\mathcal{Q}_0$ and $k' = 1$, it holds that

$$\begin{aligned}|\mathrm{Pr}_1(\mathcal{Q}_0,\beta) - \mathrm{Pr}_2(\mathcal{Q}_0,\beta)| &= |\mathrm{Pr}(q_1|\{q_1\}) \times \mathrm{Pr}_\sigma(q_1,\beta) - \mathrm{Pr}(q_2|\{q_2\}) \times \mathrm{Pr}_\sigma(q_2,\beta)| \\ &= 0.1 \leq \epsilon; \\ |\mathrm{Pr}_1(\mathcal{Q}_0,\lambda) - \mathrm{Pr}_2(\mathcal{Q}_0,\lambda)| &= |\mathrm{Pr}(q_1|\{q_1\}) \times \mathrm{Pr}_\sigma(q_1,\lambda) - \mathrm{Pr}(q_2|\{q_2\}) \times \mathrm{Pr}_\sigma(q_2,\lambda)| \\ &= 0.3 > \epsilon; \\ |\mathrm{Pr}_1(\mathcal{Q}_0,\gamma) - \mathrm{Pr}_2(\mathcal{Q}_0,\gamma)| &= |\mathrm{Pr}(q_2|\{q_2\}) \times \mathrm{Pr}_\sigma(q_2,\gamma)| = 0.4 > \epsilon.\end{aligned}$$

Two systems $G(q_1)$ and $G(q_2)$ do not satisfy ϵ-state differential privacy with $\epsilon = 0.12$, within three-step observation extensions, after the systems generate the empty string from q_1 and q_2, respectively.

Theorem 1. *Two systems satisfy ϵ-state differential privacy, within $k \in \mathbb{N}^+$-step observation extensions, after the systems generate a given observation from two given adjacent initial states if, and only if Algorithm 2 returns true.*

Proof. (if) Given two probabilistic automata $G(q_0) = (Q, \Sigma, \delta, q_0, \rho)$ and $G(q_0') = (Q, \Sigma, \delta, q_0', \rho)$, a positive real number ϵ, and an observation $\omega \in \Sigma_o^*$, let $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$ be the verifier due to Algorithm 1. Given a positive integer $k \in \mathbb{N}^+$, for any $k' \leq k$ with $k' \in \mathbb{N}^+$, and for any observation $\omega' \in \Sigma_o^*$ whose length is equal to k' generated from $\mathcal{Q}_0$, $\mathrm{Pr}_1(\mathcal{Q}_0, \omega')$ and $\mathrm{Pr}_2(\mathcal{Q}_0, \omega')$ in Algorithm 2 are the probabilities of generating ω' after the systems generate ω from q_0 and q_0', respectively. For any $\omega' \in \Sigma_o^*$ generated within k-step observation extensions after the systems generate ω from q_0 and q_0', the difference between $\mathrm{Pr}_1(\mathcal{Q}_0, \omega')$ and $\mathrm{Pr}_2(\mathcal{Q}_0, \omega')$ is less than or equal to ϵ if Algorithm 2 returns true, that is, for all $k' \leq k$ and all $\omega' \in \mathcal{L}_o(q_0, \omega, k') \cup \mathcal{L}_o(q_0', \omega, k')$, it holds $|\mathrm{Pr}_o(q_0, \omega, k', \omega') - \mathrm{Pr}_o(q_0', \omega, k', \omega')| \leq \epsilon$. The two systems $G(q_0)$ and $G(q_0')$ satisfy ϵ-state differential privacy, within k-step observation extensions, after the systems generate a given observation ω from two given adjacent initial states q_0 and q_0'.

(only if) If Algorithm 2 returns false, there exists an observation $\omega' \in \Sigma_o^*$ such that the difference between $\mathrm{Pr}_1(\mathcal{Q}_0, \omega')$ and $\mathrm{Pr}_2(\mathcal{Q}_0, \omega')$ is larger than ϵ, that is, there exists a positive integer $k' \leq k$ such that $|\mathrm{Pr}_o(q_0, \omega, k', \omega') - \mathrm{Pr}_o(q_0', \omega, k', \omega')| > \epsilon$ holds, where $\omega' \in \mathcal{L}_o(q_0, \omega, k') \cup \mathcal{L}_o(q_0', \omega, k')$. The two systems $G(q_0)$ and $G(q_0')$ do not satisfy ϵ-state differential privacy. This reveals that Algorithm 2 returns true if two systems $G(q_0)$ and $G(q_0')$ satisfy ϵ-state differential privacy, within k-step observation extensions, after $G(q_0)$ and $G(q_0')$ generate a given observation from two given adjacent initial states q_0 and q_0'. □

5. Supervisory Control for Enforcing State Differential Privacy

As seen from Section 4, if the probability distributions of generating observations within a given finite step observation extension, after two systems generating a given observation from two adjacent initial states are approximate, then the two systems satisfy state differential privacy. To ensure that the two systems satisfy state differential privacy within a given finite step observation extension, we present a supervisory control strategy for enforcing state differential privacy.

Given a matrix M, we use $M[i][j]$ to describe the element in the i-th row and j-th column of M, where $i, j \in \mathbb{N}^+$. The number of rows or columns in M is represented as $\mathbb{N}^r(M) \in \mathbb{N}^+$ or $\mathbb{N}^c(M) \in \mathbb{N}^+$. Moreover, $M[:][j]$ and $M[i][:]$ are the j-th column and the i-th row vectors of M, respectively. Moreover, if M is a row (column) vector, $M[i]$ is the element in the i-th column (row) of M. Given two $m \times n$ matrices M_1 and M_2, an $m \times 2n$ matrix $M = [M_1|M_2]$ is a horizontal extension of M_1 and M_2.

Given a verifier $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$ and a state $\mathcal{Q} \in Q_v$, the set $\mathcal{C}(\mathcal{Q})$ is defined as

$$\mathcal{C}(\mathcal{Q}) = \{e \in \Sigma_o \mid (\exists \mathcal{Q}' \in \mathcal{Q}^\sharp \setminus \{\mathcal{Q}\})\ \delta_v(\mathcal{Q}, e) = \mathcal{Q}'\}.$$

A mapping $H : \Sigma_o \to \mathbb{N}^+$ assigns to an observable event a unique positive integer. For a state $\mathcal{Q} \in Q_v$, we sort all events $e \in \mathcal{C}(\mathcal{Q})$ by $H(e)$ from small to large, and then give e an index value $I(\mathcal{Q}, e)$, where $I : Q_v \times \Sigma_o \to \mathbb{N}^+$ is a mapping. For two states $\mathcal{Q} = \mathcal{S}_1 \times \mathcal{S}_2 \in Q_v$ and $\mathcal{Q}' = \mathcal{S}_1' \times \mathcal{S}_2' \in {}^\natural\mathcal{Q}$, the set of all enabled events from $\mathcal{Q}'$ to $\mathcal{Q}$ is denoted as $\mathcal{E}(\mathcal{Q}', \mathcal{Q}) = \{e \in \Sigma_o \mid \delta_v(\mathcal{Q}', e) = \mathcal{Q}\}$. For any event $e \in \mathcal{E}(\mathcal{Q}', \mathcal{Q})$, $X(\mathcal{Q}', \mathcal{Q}, e)$ is defined as a $|\mathcal{C}(\mathcal{Q}')|$-dimensional column vector, which contains zeros and ones only. We associate a binary scalar $X(\mathcal{Q}', \mathcal{Q}, e)[v]$ defined as follows:

$$X(\mathcal{Q}', \mathcal{Q}, e)[v] = \begin{cases} 1, & v = I(\mathcal{Q}', e) \\ 0, & v \neq I(\mathcal{Q}', e) \end{cases}$$

where $1 \leq v \leq |\mathcal{C}(\mathcal{Q}')|$ and $v \in \mathbb{N}^+$. $X(\mathcal{Q}', \mathcal{Q})$ is a horizontal extension of $X(\mathcal{Q}', \mathcal{Q}, e)$ for all $e \in \mathcal{E}(\mathcal{Q}', \mathcal{Q})$ sorted by increasing the value of $I(\mathcal{Q}', e)$.

Example 7. *A probabilistic automaton structure is shown in Figure 3. Given two adjacent initial states q_1 and q_2, suppose that $\Sigma_o = \{\alpha, \beta, \gamma, \lambda, \mu\}$ and $\Sigma_{uo} = \{\tau\}$. A verifier V_ε is shown in Figure 4. Let $H(\alpha) = 1$, $H(\beta) = 2$, $H(\gamma) = 3$, $H(\lambda) = 4$ and $H(\mu) = 5$. For initial state $\mathcal{Q}_0$, $\mathcal{C}(\mathcal{Q}_0) = \{\gamma, \beta, \lambda\}$ holds. Since $I(\mathcal{Q}_0, \beta) = 1$, $I(\mathcal{Q}_0, \gamma) = 2$ and $I(\mathcal{Q}_0, \lambda) = 3$, it holds that $\mathcal{E}(\mathcal{Q}_0, \mathcal{Q}_2) = \{\beta, \lambda\}$ and $X(\mathcal{Q}_0, \mathcal{Q}_2) = [X_\beta | X_\lambda]$, where $X_\beta = (1, 0, 0)^T$ and $X_\lambda = (0, 0, 1)^T$.*

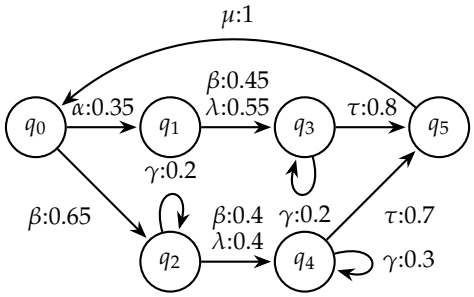


Figure 3. A probabilistic automaton structure G_s'.

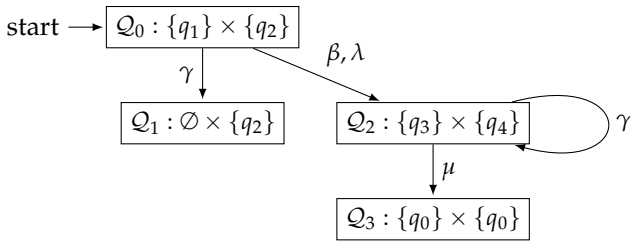


Figure 4. Verifier V_ε for $G'(q_1)$ and $G'(q_2)$.

Given a probabilistic automaton structure $G_s = (Q, \Sigma, \delta, \rho)$ with two adjacent initial states q_0, q_0' and an observation $\omega \in \Sigma_o^*$, let $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$ be the verifier. For a state $\mathcal{Q} = \mathcal{S}_1 \times \mathcal{S}_2 \in Q_v$ and $\delta_v(\mathcal{Q}_0, \omega') = \mathcal{Q}$, $Z_{\omega'}(\mathcal{S}_1 | \mathcal{Q})$ and $Z_{\omega'}(\mathcal{S}_2 | \mathcal{Q})$ are two $|\mathcal{C}(\mathcal{Q})|$-dimensional row vectors. For any $e \in \mathcal{C}(\mathcal{Q})$, $I(\mathcal{Q}, e) \in \{1, 2, \ldots, |\mathcal{C}(\mathcal{Q})|\}$ and $i \in \{1, 2\}$, it holds that

$$Z_{\omega'}(\mathcal{S}_1 | \mathcal{Q})[I(\mathcal{Q}, e)] = \sum_{q \in \mathcal{S}_1} \sum_{s \in \sigma(q, e)} [\Pr(q | \varphi(q_0, \omega\omega')) \times \Pr_\sigma(q, s)];$$

$$Z_{\omega'}(\mathcal{S}_2 | \mathcal{Q})[I(\mathcal{Q}, e)] = \sum_{q \in \mathcal{S}_2} \sum_{s \in \sigma(q, e)} [\Pr(q | \varphi(q_0', \omega\omega')) \times \Pr_\sigma(q, s)].$$

Example 8. *Consider the verifier in Figure 4. For state $\mathcal{Q}_0$, $\mathcal{C}(\mathcal{Q}_0) = \{\gamma, \beta, \lambda\}$ holds. Since $I(\mathcal{Q}_0, \beta) = 1$, $I(\mathcal{Q}_0, \gamma) = 2$ and $I(\mathcal{Q}_0, \lambda) = 3$, we have $Z_\varepsilon(\{q_1\} | \mathcal{Q}_0) = (0.45, 0, 0.55)$ and $Z_\varepsilon(\{q_2\} | \mathcal{Q}_0) = (0.4, 0.2, 0.4)$. For state $\mathcal{Q}_2$, $\mathcal{C}(\mathcal{Q}_2) = \{\mu\}$ holds. We have $Z_{\omega'}(\{q_3\} | \mathcal{Q}_2) = \Pr_\sigma(q_3, \tau\mu) = 0.8$ and $Z_{\omega'}(\{q_4\} | \mathcal{Q}_2) = \Pr_\sigma(q_4, \tau\mu) = 0.7$, where $\omega' \in \{\beta, \lambda\}$.*

Given a probabilistic automaton structure $G_s = (Q, \Sigma, \delta, \rho)$ with two adjacent initial states q_0, q_0' and an observation $\omega \in \Sigma_o^*$, let $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$ be the verifier. For a state $\mathcal{Q} = \mathcal{S}_1 \times \mathcal{S}_2 \in Q_v$ with $\delta_v(\mathcal{Q}_0, \omega') = \mathcal{Q}$, $X^{\omega'}(\mathcal{S}_i|\mathcal{Q})$ is called a *probability matrix* for $\mathcal{S}_i$, $\mathcal{Q}$ and ω', where $i \in \{1, 2\}$, respectively, defined by

1. if ${}^\sharp\mathcal{Q} = \emptyset$, it holds $X^\varepsilon(\mathcal{S}_i|\mathcal{Q}) = Z_\varepsilon(\mathcal{S}_i|\mathcal{Q})$;
2. if ${}^\sharp\mathcal{Q} \neq \emptyset$, for $\mathcal{Q}' = \mathcal{S}_1' \times \mathcal{S}_2' \in {}^\sharp\mathcal{Q}$ with $\delta_v(\mathcal{Q}_0, \omega'') = \mathcal{Q}'$, $\delta_v(\mathcal{Q}', e) = \mathcal{Q}$, and $\omega'' e = \omega'$, it holds that

$$X_m^{\omega'}(\mathcal{S}_i|\mathcal{Q}) = [X^{\omega''}(\mathcal{S}_i'|\mathcal{Q}') \times X(\mathcal{Q}', \mathcal{Q})]^T[:][m] \times Z_{\omega'}(\mathcal{S}_i|\mathcal{Q}).$$

Then

$$X^{\omega'}(\mathcal{S}_i|\mathcal{Q}) = [X_1^{\omega'}(\mathcal{S}_i|\mathcal{Q})^T| \ldots |X_n^{\omega'}(\mathcal{S}_i|\mathcal{Q})^T]^T,$$

where $n = \mathbb{N}^r(X^{\omega''}(\mathcal{S}_i'|\mathcal{Q}'))$ and $m \in \{1, 2, \ldots, n\}$.

The computation of probability matrices for a state in a verifier is implemented by Algorithm 3, whose complexity is $\mathcal{O}(|\mathcal{Q}_v| \times 2^{|Q|})$.

Example 9. *Consider the verifier in Figure 4. For initial state $\mathcal{Q}_0$, $X^\varepsilon(\{q_1\}|\mathcal{Q}_0) = (0.45, 0, 0.55)$ and $X^\varepsilon(\{q_2\}|\mathcal{Q}_0) = (0.4, 0.2, 0.4)$ hold. Given state $\mathcal{Q}_2$, we have*

$$X^\varepsilon(\{q_1\}|\mathcal{Q}_0) \times X(\mathcal{Q}_0, \mathcal{Q}_2) = (0.45, 0.55); \;\; X^\varepsilon(\{q_2\}|\mathcal{Q}_0) \times X(\mathcal{Q}_0, \mathcal{Q}_2) = (0.4, 0.4).$$

For $\omega' \in \{\beta, \lambda\}$, we have

$$X^{\omega'}(\{q_3\}|\mathcal{Q}_2) = (0.45, 0.55)^T[:][1] \times Z_{\omega'}(\{q_3\}|\mathcal{Q}_2) = (0.36, 0.44)^T;$$
$$X^{\omega'}(\{q_4\}|\mathcal{Q}_2) = (0.4, 0.4)^T[:][1] \times Z_{\omega'}(\{q_4\}|\mathcal{Q}_2) = (0.28, 0.28)^T.$$

As in the classical supervisory control theory of DESs, the set Σ is partitioned into Σ_c and Σ_{uc} ($\Sigma = \Sigma_c \dot\cup \Sigma_{uc}$), the sets of controllable and uncontrollable events, respectively. Traditionally, we can only disable controllable events $e \in \Sigma_c$. This paper assumes that all observable events are controllable and all unobservable events are uncontrollable.

For a probabilistic automaton $G = (Q, \Sigma, \delta, q_0, \rho)$, the probabilities of the remaining enabled events proportionally increase after a controllable event is disabled. If a controllable event is disabled at state $q \in Q$, the set of all enabled events at q is updated. The firing probability of a remaining enabled event e at q is equal to $\rho(q, e) = \rho(q, e)/\sum_{e' \in \mathcal{E}(q)} \rho(q, e')$.

Example 10. *Consider the probabilistic automaton structure in Figure 3. If event β is disabled at state q_2, the plant can choose between γ and λ to occur at q_2. The firing probabilities of γ and λ at q_2 are $\rho(q_2, \gamma) = \rho(q_2, \gamma)/(\rho(q_2, \gamma) + \rho(q_2, \lambda)) = 1/3$ and $\rho(q_2, \lambda) = \rho(q_2, \lambda)/(\rho(q_2, \gamma) + \rho(q_2, \lambda)) = 2/3$, respectively.*

Algorithm 3: Computation of probability matrices for a state in a verifier

Input: A verifier $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$ and a state $\mathcal{Q} = \mathcal{S}_1 \times \mathcal{S}_2$ with $\delta_v(\mathcal{Q}_0, \omega') = \mathcal{Q}$

Output: Probability matrices $X^{\omega'}(\mathcal{S}_1|\mathcal{Q})$ and $X^{\omega'}(\mathcal{S}_2|\mathcal{Q})$

```
1  ♯Q ← ∅;
2  foreach Q_a ∈ Q_v do
3      if δ_v(Q_0, ω'') = Q_a & δ_v(Q_a, e) = Q & ω''e = ω' then
4          ♯Q ← ♯Q ∪ {Q_a};
5  foreach Q' ← S'_1 × S'_2 ∈ ♯Q do
6      X(Q', Q) ← []; C(Q') ← ∅; C(Q) ← ∅;
7      foreach e ∈ Σ_o & Q_a ∈ {Q', Q} do
8          if δ_v(Q_a, e)! & δ_v(Q_a, e) ≠ Q_a then
9              C(Q_a) ← C(Q_a) ∪ {e};
10     foreach e ∈ Σ_o & δ_v(Q', e) = Q do
11         for a = 1, a ≤ |C(Q')|; a++ do
12             if a = I(Q', e) then
13                 X(Q', Q, e)[a][1] ← 1;
14             else
15                 X(Q', Q, e)[a][1] ← 0;
16         X(Q', Q) ← [X(Q', Q)|X(Q', Q, e)];
17     foreach i ∈ {1, 2} do
18         Y_i ← X^{ω''}(S'_i|Q') × X(Q', Q);
19     obtain φ(q_0, ωω'), φ(q'_0, ωω') by Algorithm 1;
20     foreach e ∈ C(Q) do
21         foreach q ∈ S_1 do
22             compute σ(q, e) by Algorithm 2;
23         Z_{ω'}(S_1|Q)[1][I(Q, e)] ← Σ_{s∈σ(q,e)} Σ_{q∈S_1} [Pr(q|φ(q_0, ωω')) × Pr_σ(q, s)];
24         foreach q ∈ S_2 do
25             compute σ(q, e) by Algorithm 2;
26         Z_{ω'}(S_2|Q)[1][I(Q, e)] ← Σ_{s∈σ(q,e)} Σ_{q∈S_2} [Pr(q|φ(q'_0, ωω')) × Pr_σ(q, s)];
27     foreach i ∈ {1, 2} do
28         n ← ℕ^r(X^{ω''}(S'_i|Q'));
29         for m = 1, m ≤ n; m++ do
30             X^{ω'}_m(S_i|Q) ← Y_i^T[:][m] × Z_{ω'}(S_i|Q);
31         X^{ω'}(S_i|Q) = [X^{ω'}_1(S_i|Q)^T| ... |X^{ω'}_n(S_i|Q)^T]^T;
```

A function $\mathcal{D} : Q \to 2^\Sigma$ is a mapping that assigns to a state in a probabilistic automaton $G = (Q, \Sigma, \delta, q_0, \rho)$ a set of disabled events. Algorithm 4 computes a control specification, which is to disable certain events at specific states. Given a probabilistic automaton structure $G_s = (Q, \Sigma, \delta, \rho)$ with two adjacent initial states q_0, q'_0 and an observation $\omega \in \Sigma_o^*$, let $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$ be the verifier. Given a positive integer $k \in \mathbb{N}^+$, for any $k' \leq k$ with $k' \in \mathbb{N}^+$, Q_n is the set of reachable states with k'-step observation extensions. For any state $\mathcal{Q} = \mathcal{S}_1 \times \mathcal{S}_2 \in Q_n$ with $\delta_v(\mathcal{Q}_0, \omega') = \mathcal{Q}$, and for any two numbers x_1, x_2 at the same position in $X^{\omega'}(\mathcal{S}_1|\mathcal{Q})$, $X^{\omega'}(\mathcal{S}_2|\mathcal{Q})$, we need to decide whether $|x_1 - x_2|$ is larger than ϵ. If so, event e is disabled at all $q \in \mathcal{S}_1 \cup \mathcal{S}_2$, where $I(\mathcal{Q}, e) = j$ and x_1 is in the j-th column of $X^{\omega'}(\mathcal{S}_1|\mathcal{Q})$. $X^{\omega'}(\mathcal{S}_1|\mathcal{Q})$ and $X^{\omega'}(\mathcal{S}_2|\mathcal{Q})$ are then updated. Moreover, for any observable event e with $\delta_v(\mathcal{Q}, e) = \mathcal{Q}$, w_1 and w_2 are the probabilities of generating e from

$q \in \mathcal{S}_1$ and $q' \in \mathcal{S}_2$, respectively. For any two numbers x_1, x_2 at the same position in $X^{\omega'}(\mathcal{S}_1|\mathcal{Q})$, $X^{\omega'}(\mathcal{S}_2|\mathcal{Q})$, and for any positive integer $n \leq k - k'$ with $n \in \mathbb{N}^+$, we decide whether $|w_1^n \times x_1 - w_2^n \times x_2| \leq \epsilon$ holds until $w_1^n \times x_1$ and $w_2^n \times x_2$ are both less than or equal to ϵ. If $|w_1^n \times x_1 - w_2^n \times x_2| > \epsilon$, event e is disabled at all $q \in \mathcal{S}_1 \cup \mathcal{S}_2$. Its complexity is $\mathcal{O}(k \times |Q_v|^2 \times |\Sigma_o| \times 2^{|\mathcal{Q}|})$.

Algorithm 4: Computation of a control specification

Input: A verifier $V_\omega = (Q_v, \Sigma_o, \delta_v, \mathcal{Q}_0)$ and a positive integer k
Output: The function $\mathcal{D}$

1 $Q_n \leftarrow \{\mathcal{Q}_0\}$; $Q_m \leftarrow \emptyset$; $\forall q \in Q : \mathcal{D}(q) \leftarrow \emptyset$;
2 **for** $k' = 1, k' \leq k; k'++$ **do**
3 **foreach** $\mathcal{Q} = \mathcal{S}_1 \times \mathcal{S}_2 \in Q_n$ & $\delta_v(\mathcal{Q}_0, \omega') = \mathcal{Q}$ **do**
4 $M \leftarrow X^{\omega'}(\mathcal{S}_1|\mathcal{Q}) - X^{\omega'}(\mathcal{S}_2|\mathcal{Q})$; $\mathcal{Q}^\sharp \leftarrow \emptyset$;
5 **foreach** $M[i][j]$ **do**
6 **if** $|M[i][j]| > \epsilon$ **then**
7 **foreach** $q \in \mathcal{S}_1 \cup \mathcal{S}_2$ & $I(\mathcal{Q}, e) = j$ **do**
8 $\mathcal{D}(q) \leftarrow \mathcal{D}(q) \cup \{e\}$;
9 update $X^{\omega'}(\mathcal{S}_1|\mathcal{Q})$, $X^{\omega'}(\mathcal{S}_2|\mathcal{Q})$;
10 go to line 5;
11 obtain $\varphi(q_0, \omega\omega')$, $\varphi(q_0', \omega\omega')$ by Algorithm 1;
12 **foreach** $e \in \Sigma_o$ & $\delta_v(\mathcal{Q}, e) = \mathcal{Q}$ **do**
13 $w_1 \leftarrow \sum_{q \in \mathcal{S}_1}[\Pr(q|\varphi(q_0, \omega\omega')) \times \rho(q, e)]$;
14 $w_2 \leftarrow \sum_{q \in \mathcal{S}_2}[\Pr(q|\varphi(q_0', \omega\omega')) \times \rho(q, e)]$;
15 **for** $n = 1, n \leq k - k'; n++$ **do**
16 **foreach** $x_1 = X^{\omega'}(\mathcal{S}_1|\mathcal{Q})[i][j]$ & $x_2 = X^{\omega'}(\mathcal{S}_2|\mathcal{Q})[i][j]$ **do**
17 **if** $w_1^n \times x_1 \leq \epsilon$ & $w_2^n \times x_2 \leq \epsilon$ **then**
18 break;
19 **if** $|w_1^n \times x_1 - w_2^n \times x_2| > \epsilon$ **then**
20 **foreach** $q \in \mathcal{S}_1 \cup \mathcal{S}_2$ **do**
21 $\mathcal{D}(q) \leftarrow \mathcal{D}(q) \cup \{e\}$;
22 go to line 12;
23 **foreach** $e \in \Sigma_o$ & $\delta_v(\mathcal{Q}, e)!$ & $\delta_v(\mathcal{Q}, e) \neq \mathcal{Q}$ **do**
24 $\mathcal{Q}^\sharp \leftarrow \mathcal{Q}^\sharp \cup \{\delta_v(\mathcal{Q}, e)\}$;
25 $Q_n \leftarrow Q_n \setminus \{\mathcal{Q}\}$; $Q_m \leftarrow Q_m \cup (\mathcal{Q}^\sharp \setminus \{\mathcal{Q}\})$;
26 $Q_n \leftarrow Q_m$; $Q_m \leftarrow \emptyset$;

A supervisory control function for a probability automaton $G = (Q, \Sigma, \delta, q_0, \rho)$ is $\mathcal{V} : Q \to 2^\Sigma$ that assigns to a state in G a set of enabled events, where $\mathcal{V}(q) = \mathcal{E}(q) \setminus \mathcal{D}(q)$ for all $q \in Q$. The next event allowed to happen at q by supervisory control is $e \in \mathcal{V}(q)$. A supervisor implementing the supervisory control function $\mathcal{V}$ can be constructed if $\bigcup_{q \in Q} \mathcal{D}(q) \cap \Sigma_{uc} = \emptyset$ holds.

Example 11. *Let us consider the verifier in Figure 4. Suppose* $\Sigma_c = \{\alpha, \beta, \lambda, \gamma, \mu\}$, $\Sigma_{uc} = \{\tau\}$, $k = 10$ *and* $\epsilon = 0.12$. *For* $k' = 1$, $X^\epsilon(\{q_1\}|\mathcal{Q}_0) = (0.45, 0, 0.55)$ *and* $X^\epsilon(\{q_2\}|\mathcal{Q}_0) = (0.4, 0.2, 0.4)$ *hold. Since* $|0 - 0.2| > \epsilon$, *disable event* γ *at state* q_2. *We update* $X^\epsilon(\{q_1\}|\mathcal{Q}_0) = (0.45, 0.55)$ *and* $X^\epsilon(\{q_2\}|\mathcal{Q}_0) = (0.5, 0.5)$. *For* $k' = 2$, $X^{\omega'}(\{q_3\}|\mathcal{Q}_2) = (0.36, 0.44)^T$ *and* $X^{\omega'}(\{q_4\}|\mathcal{Q}_2) = (0.35, 0.35)^T$ *hold, where* $\omega' \in \{\beta, \lambda\}$. *Since* $\rho(q_3, \gamma) \times X^{\omega'}(\{q_3\}|\mathcal{Q}_2) = (0.072, 0.088)^T$ *and* $\rho(q_4, \gamma) \times X^{\omega'}(\{q_4\}|\mathcal{Q}_2) = (0.105, 0.105)^T$, *event* γ *is not disabled at states* q_3 *and* q_4. *For* $2 \leq k' \leq k$ *and* $k' \in \mathbb{N}^+$, *since* $\mathcal{Q}_3^\sharp = \emptyset$, *we do not need to do more analysis. The control specification is that event* γ *is disabled at state* q_2, *that is,* $\mathcal{D}(q_2) = \{\gamma\}$. *Due to*

$\bigcup_{q\in Q}\mathcal{D}(q)\cap\Sigma_{uc}=\emptyset$, *there exists a supervisor to control the system behavior such that* $G'(q_1)$ *and* $G'(q_2)$ *satisfy* ϵ*-state differential privacy, within* 10*-step observation extensions, after* $G'(q_1)$ *and* $G'(q_2)$ *generate the empty string from* q_1 *and* q_2*. The skeleton of a supervisor implementing* $\mathcal{V}$ *is shown in Figure 5.*

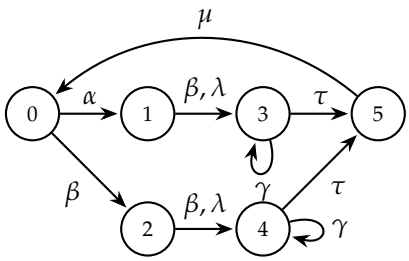


Figure 5. The skeleton of a supervisor for $G'(q_1)$ and $G'(q_2)$.

Theorem 2. *Given a probabilistic automaton structure* $G_s=(Q,\Sigma,\delta,\rho)$ *and two adjacent initial states* q_0, q_0'*,* $G(q_0)$ *and* $G(q_0')$ *controlled by the control specification due to Algorithm 4 satisfy* ϵ*-state differential privacy, within a given* k*-step observation extension, after the systems generate a given observation* $\omega\in\Sigma_o^*$ *from* q_0 *and* q_0'*.*

Proof. Given a probabilistic automaton structure $G_s=(Q,\Sigma,\delta,\rho)$ with two adjacent initial states q_0, q_0' and an observation $\omega\in\Sigma_o^*$, let $V_\omega=(Q_v,\Sigma_o,\delta_v,\mathcal{Q}_0)$ be the verifier. For $\mathcal{Q}=\mathcal{S}_1\times\mathcal{S}_2\in Q_v$ that can be reached with k'-step observation extensions and $\delta_v(\mathcal{Q}_0,\omega')=\mathcal{Q}$, where $k'\le k$ and $k'\in\mathbb{N}^+$, if there exists no observable event $e\in\Sigma_o$ such that $\delta_v(\mathcal{Q},e)=\mathcal{Q}$, and if $|X^{\omega'}(\mathcal{S}_1|\mathcal{Q})[i][j]-X^{\omega'}(\mathcal{S}_2|\mathcal{Q})[i][j]|\le\epsilon$ holds for all $i,j\in\mathbb{N}^+$, then the difference between the probabilities of generating $\omega\omega' e$ for any $e\in\Sigma_o$ from q_0 and q_0' is less than or equal to ϵ.

For $\mathcal{Q}=\mathcal{S}_1\times\mathcal{S}_2\in Q_v$ that can be reached with k'-step observation extensions and $\delta_v(\mathcal{Q}_0,\omega')=\mathcal{Q}$, if there exists an observable event $e\in\Sigma_o$ such that $\delta_v(\mathcal{Q},e)=\mathcal{Q}$, then the relation $|w_1^n\times x_1-w_2^n\times x_2|\le\epsilon$ is true until $w_1^n\times x_1\le\epsilon$ and $w_2^n\times x_2\le\epsilon$ hold for all $1\le n\le k-k'$ and $n\in\mathbb{N}^+$, where w_1 and w_2 are the probabilities of generating e from $q\in\mathcal{S}_1$ and $q'\in\mathcal{S}_2$, respectively, and x_1, x_2 are any two numbers at the same position in $X^{\omega'}(\mathcal{S}_1|\mathcal{Q}), X^{\omega'}(\mathcal{S}_2|\mathcal{Q})$, respectively. The difference between the probabilities of generating $\omega\omega'\omega_a$ for all $\omega_a\in\Sigma_o^*$ containing e from q_0 and q_0' is less than or equal to ϵ. For all $k'\in\mathbb{N}^+$ and $\omega'\in\Sigma_o^*$, where $k'\le k$ and $|\omega'|=k'$, it holds that $|\mathrm{Pr}_o(q_0,\omega,k,\omega')-\mathrm{Pr}_o(q_0',\omega,k,\omega')|\le\epsilon$. Two systems $G(q_0)$ and $G(q_0')$ controlled by the control specification due to Algorithm 4 satisfy ϵ-state differential privacy, within a given k-step observation extension, after the systems generate a given observation ω from q_0 and q_0'. □

The supervisory control is maximally permissive for ϵ-state differential privacy enforcement if the number of enabled controllable events at any state in the probabilistic automaton controlled via the supervisor is the largest compared with other supervisory control methods. If a supervisor implementing the proposed supervisory control function can be constructed, then the maximally permissive supervisory control exists.

Proposition 2. *The supervisory control under the control specification due to Algorithm 4 is maximally permissive.*

Proof. Given a probabilistic automaton structure $G_s=(Q,\Sigma,\delta,\rho)$ with two adjacent initial states q_0, q_0' and an observation $\omega\in\Sigma_o^*$, let $V_\omega=(Q_v,\Sigma_o,\delta_v,\mathcal{Q}_0)$ be the verifier. A supervisor is constructed by the control specification due to Algorithm 4. The supervisory control function for a probability automaton $G=(Q,\Sigma,\delta,q_0,\rho)$ under the control specification is $\mathcal{V}:Q\to 2^\Sigma$, where $\mathcal{V}(q)=\mathcal{E}(q)\setminus\mathcal{D}(q)$ for all $q\in Q$. Suppose that $G(q_0)$ and $G(q_0')$ reach states q and q' by generating $\omega\omega'\in\Sigma_o^*$ from q_0 and q_0' within $k\in\mathbb{N}^+$-step observation extensions, respectively. For all $e\in\mathcal{V}(q)$ (or $e\in\mathcal{V}(q')$), it holds that

$|\mathrm{Pr}_o(q_0, \omega\omega' e) - \mathrm{Pr}_o(q_0', \omega\omega' e)| \leq \epsilon$. If any event $e \in \mathcal{D}(q) \cup \mathcal{D}(q')$ occurs from q or q', we obtain $|\mathrm{Pr}_o(q_0, \omega\omega' e) - \mathrm{Pr}_o(q_0', \omega\omega' e)| > \epsilon$. Since $\mathcal{D}(q) \cup \mathcal{D}(q') \subseteq \Sigma_o$, there exists an observation $\omega\omega' e$ such that the difference between firing probabilities of $\omega\omega' e$ at q_0 and q_0' is larger than ϵ. Two systems $G(q_0)$ and $G(q_0')$ do not satisfy ϵ-state differential privacy, within k-step observation extensions, after the systems generate ω from q_0 and q_0'. We conclude that the supervisory control under the control specification due to Algorithm 4 is maximally permissive. □

6. Numerical Examples

To verify the correctness and effectiveness of the method in this paper, an experimental study in the MATLAB environment is conducted to illustrate that the proposed method achieves state differential privacy in the considered class of probabilistic automata and protects the information of initial system resource configuration.

A probabilistic automaton structure G_s is shown in Figure 6. Suppose $\Sigma_o = \{\alpha, \beta, \lambda, \mu\}$ and $\Sigma_{uo} = \{\tau\}$. Given two adjacent initial states q_0 and q_1, a verifier V_ε for $G(q_0)$ and $G(q_1)$ is shown in Figure 7. Let $H(\alpha) = 1$, $H(\beta) = 2$, $H(\lambda) = 3$ and $H(\mu) = 4$.

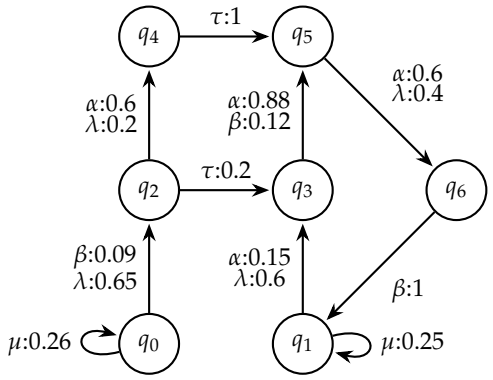


Figure 6. A probabilistic automaton structure G_s''.

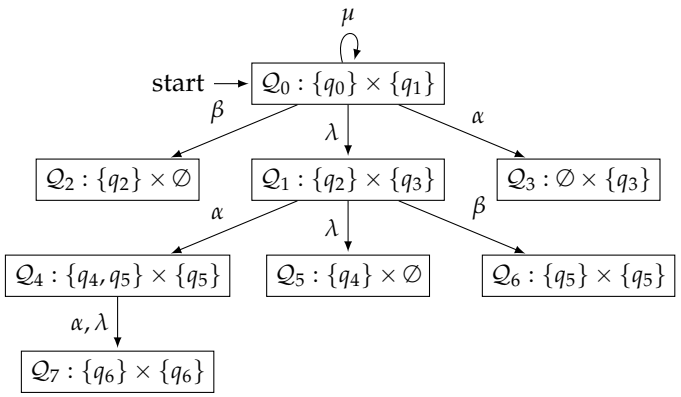


Figure 7. Verifier V_ε for $G''(q_0)$ and $G''(q_1)$.

Suppose $\Sigma_c = \{\alpha, \beta, \lambda, \mu\}$ and $\Sigma_{uc} = \{\tau\}$. Let $\epsilon = 0.1$ and $k = 5$. For $k' = 1$ and initial state $\mathcal{Q}_0$, since $I(\mathcal{Q}_0, \alpha) = 1$, $I(\mathcal{Q}_0, \beta) = 2$ and $I(\mathcal{Q}_0, \lambda) = 3$, $X^\varepsilon(\{q_0\}|\mathcal{Q}_0) = (0, 0.09, 0.65)$ and $X^\varepsilon(\{q_1\}|\mathcal{Q}_0) = (0.15, 0, 0.6)$ hold. Since $0.15 > \epsilon$, event α is disabled at states q_0 and q_1. Since $\rho(q_1, \lambda) = \rho(q_1, \lambda)/(\rho(q_1, \lambda) + \rho(q_1, \mu)) = 0.6/(0.6 + 0.25) \approx 0.7$ and $\rho(q_1, \mu) = \rho(q_1, \mu)/(\rho(q_1, \lambda) + \rho(q_1, \mu)) = 0.25/(0.6 + 0.25) \approx 0.3$, we update $X^\varepsilon(\{q_0\}|\mathcal{Q}_0) = (0.09, 0.65)$ and $X^\varepsilon(\{q_1\}|\mathcal{Q}_0) = (0, 0.7)$. Events β and λ are not disabled at states q_0 and q_1. For event μ enabled at q_0 and q_1, it holds that

$$\rho(q_0, \mu)^1 \times X^\varepsilon(\{q_0\}|\mathcal{Q}_0) = 0.26 \times (0.09, 0.65) = (0.0234, 0.169);$$
$$\rho(q_1, \mu)^1 \times X^\varepsilon(\{q_1\}|\mathcal{Q}_0) = 0.3 \times (0, 0.7) = (0, 0.21);$$

$$\rho(q_0,\mu)^2 \times X^{\varepsilon}(\{q_0\}|\mathcal{Q}_0) = 0.26^2 \times (0.09, 0.65) \approx (0.006, 0.044) \leq (0.1, 0.1);$$
$$\rho(q_1,\mu)^2 \times X^{\varepsilon}(\{q_1\}|\mathcal{Q}_0) = 0.3^2 \times (0, 0.7) = (0, 0.063) \leq (0.1, 0.1).$$

Event μ is not disabled at states q_0 and q_1. For $k' = 2$ and state $\mathcal{Q}_1$, it holds that

$$X^{\varepsilon}(\{q_0\}|\mathcal{Q}_0) \times X(\mathcal{Q}_0, \mathcal{Q}_1) = (0.09, 0.65) \times (0, 1)^T = 0.65;$$
$$X^{\varepsilon}(\{q_1\}|\mathcal{Q}_0) \times X(\mathcal{Q}_0, \mathcal{Q}_1) = (0, 0.7) \times (0, 1)^T = 0.7;$$
$$X^{\lambda}(\{q_2\}|\mathcal{Q}_1) = 0.65 \times Z_{\lambda}(\{q_2\}|\mathcal{Q}_1) = 0.65 \times (0.6 + 0.2 \times 0.88, 0.2 \times 0.12, 0.2)$$
$$= (0.5044, 0.0156, 0.13);$$
$$X^{\lambda}(\{q_3\}|\mathcal{Q}_1) = 0.7 \times Z_{\lambda}(\{q_3\}|\mathcal{Q}_1) = 0.7 \times (0.88, 0.12, 0) = (0.616, 0.084, 0).$$

Since $0.13 > \epsilon$, event λ is disabled at states q_2 and q_3. We update

$$\rho(q_2,\alpha) = \rho(q_2,\alpha)/(\rho(q_2,\alpha) + \rho(q_2,\tau)) = 0.6/(0.6 + 0.2) = 0.75;$$
$$\rho(q_2,\tau) = \rho(q_2,\tau)/(\rho(q_2,\alpha) + \rho(q_2,\tau)) = 0.2/(0.6 + 0.2) = 0.25;$$
$$X^{\lambda}(\{q_2\}|\mathcal{Q}_1) = 0.65 \times Z_{\lambda}(\{q_2\}|\mathcal{Q}_1) = 0.65 \times (0.75 + 0.25 \times 0.88, 0.25 \times 0.12)$$
$$= (0.6305, 0.0195);$$
$$X^{\lambda}(\{q_3\}|\mathcal{Q}_1) = 0.7 \times Z_{\lambda}(\{q_3\}|\mathcal{Q}_1) = 0.7 \times (0.88, 0.12) = (0.616, 0.084).$$

Events α and β are not disabled at states q_2 and q_3. For $k' = 3$ and state $\mathcal{Q}_4$, it holds

$$X^{\lambda}(\{q_2\}|\mathcal{Q}_1) \times X(\mathcal{Q}_1, \mathcal{Q}_4) = (0.6305, 0.0195) \times (1, 0)^T = 0.6305;$$
$$X^{\lambda}(\{q_3\}|\mathcal{Q}_1) \times X(\mathcal{Q}_1, \mathcal{Q}_4) = (0.616, 0.084) \times (1, 0)^T = 0.616;$$
$$\Pr(\{q_4, q_5\}|\varphi(q_0, \lambda\alpha)) = 0.65 \times 0.75/(0.65 \times 0.75 + 0.65 \times 0.25 \times 0.88) \approx 0.773;$$
$$\Pr(\{q_5\}|\varphi(q_0, \lambda\alpha)) = 0.65 \times 0.25 \times 0.88/(0.65 \times 0.75 + 0.65 \times 0.25 \times 0.88) \approx 0.227;$$
$$X^{\lambda\alpha}(\{q_4, q_5\}|\mathcal{Q}_4) = 0.6305 \times Z_{\lambda\alpha}(\{q_4, q_5\}|\mathcal{Q}_4) = 0.6305 \times$$
$$(0.773 \times 1 \times 0.6 + 0.227 \times 0.6, 0.773 \times 1 \times 0.4 + 0.227 \times 0.4) = (0.3783, 0.2522);$$
$$X^{\lambda\alpha}(\{q_5\}|\mathcal{Q}_4) = 0.616 \times Z_{\lambda\alpha}(\{q_5\}|\mathcal{Q}_4) = 0.616 \times (0.6, 0.4) = (0.3696, 0.2464).$$

Events α and λ are not disabled at states q_4 and q_5. For $k' > 3$, there is no state transition with k'-step observation extensions in V_{ε}. The control specification is that event α is disabled at state q_1 and event λ is disabled at state q_2. The skeleton of a supervisor is shown in Figure 8. For two probabilistic automata $G(q_0)$ and $G(q_1)$ controlled via the supervisor, the probability distributions of generating observations within five-step observation extensions from two adjacent initial states q_0 and q_1 are shown in Figures 9a–d and 10.

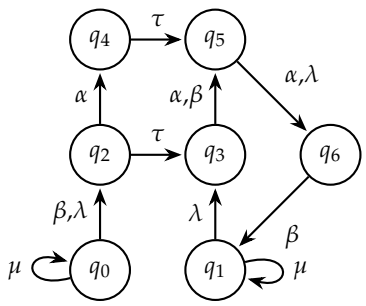


Figure 8. The skeleton of a supervisor for $G''(q_0)$ and $G''(q_1)$.

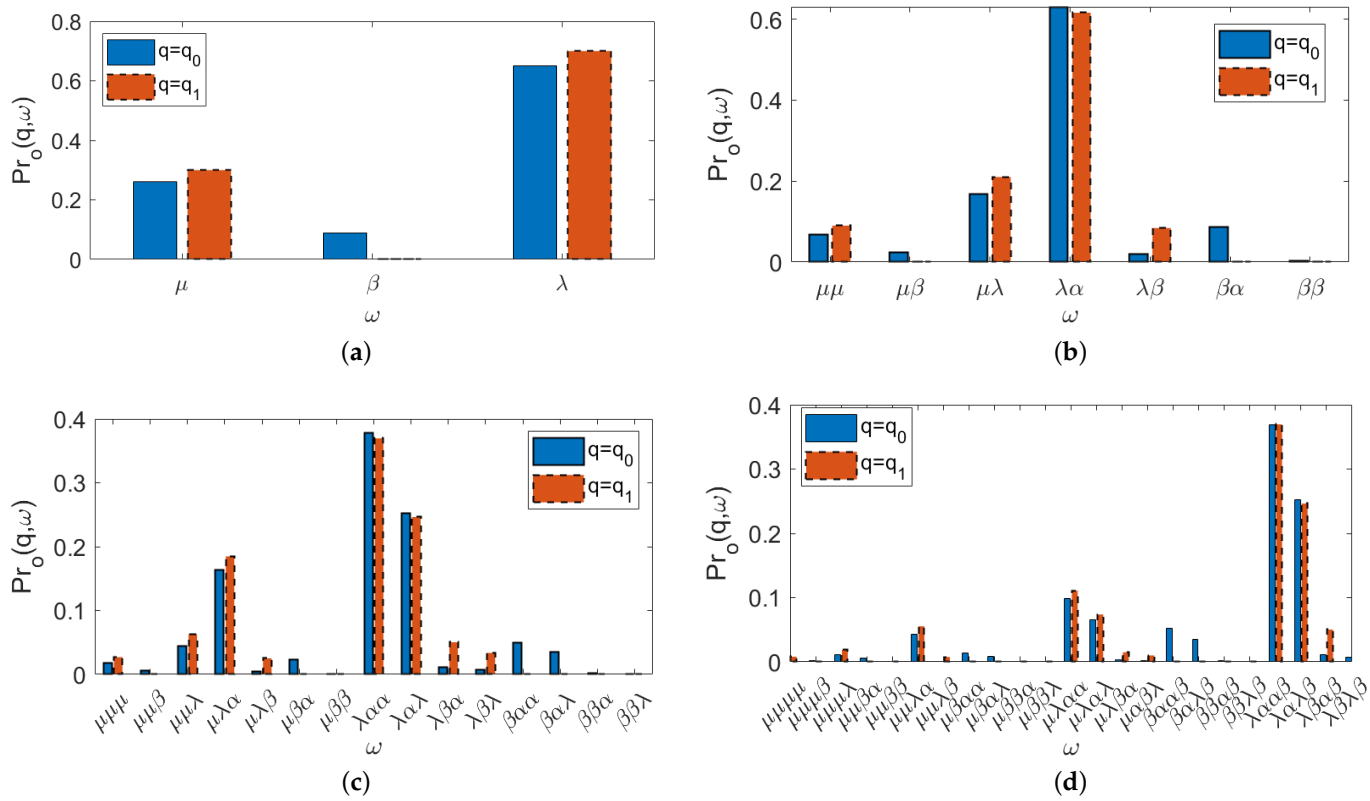


Figure 9. The probability distributions of generating observations within four-step observation extensions from q_0 and q_1. (**a**) $k = 1$; (**b**) $k = 2$; (**c**) $k = 3$; (**d**) $k = 4$.

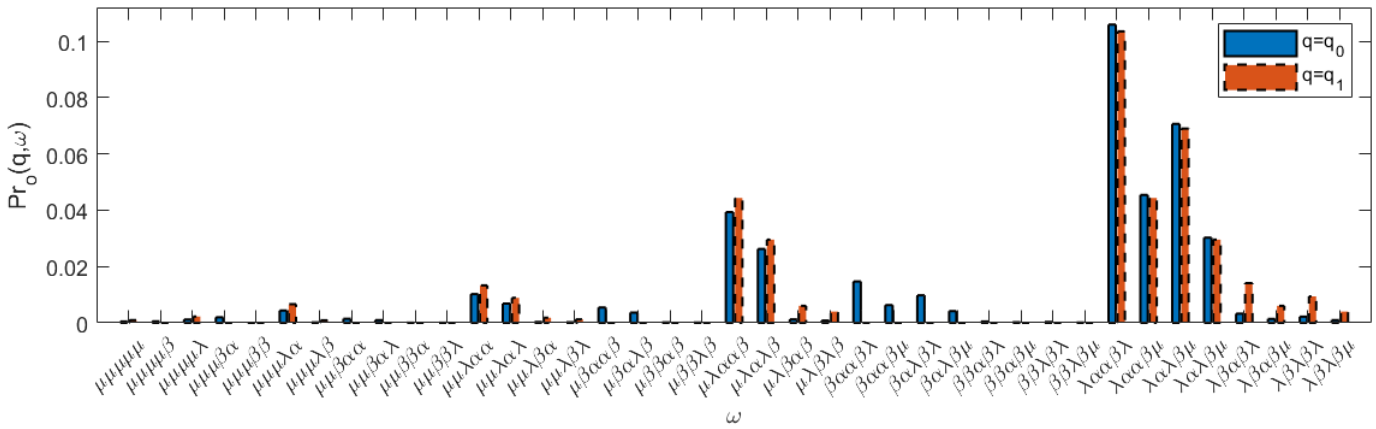


Figure 10. The probability distributions of generating observations with five-step observation extensions from q_0 and q_1.

As shown in Figures 9a–d and 10, two systems $G(q_0)$ and $G(q_1)$ controlled via the supervisor satisfy 0.1-state differential privacy, within five-step observation extensions, after the systems generate the empty string from q_0 and q_1. The smaller the ϵ is, the more events are disabled, and the fewer observations the systems generate. The supervisory control is maximally permissive. Two systems $G(q_0)$ and $G(q_1)$ controlled via the supervisor satisfy 0.1-state differential privacy while generating the most observations.

These results indicate that our method solves the problems formulated in Section 3. By constructing a verifier, we verify whether two probabilistic automata with two adjacent initial states satisfy ϵ-state differential privacy, within finite step observation extensions. Moreover, the obtained supervisor enables the two controlled systems to satisfy ϵ-state

differential privacy, while the supervisory control is maximally permissive. The existing differential privacy methods presented in probabilistic DESs cannot protect the initial state information. Our proposed method achieves state differential privacy in the considered class of probabilistic automata and protects the initial state information.

7. Conclusions

State differential privacy is defined to protect the initial state information of a DES modeled by a probabilistic automaton. The initial state information of a system represents its initial system resource configuration. Step-based state differential privacy verification is proposed in the framework of probabilistic automata. If two probabilistic automata satisfy state differential privacy, within a given finite step observation extension, after the systems generate a given observation from two adjacent initial states, then an attacker is unlikely to determine the initial states of the systems after observing the given observation; otherwise, a maximally permissive supervisory control is designed for state differential privacy enforcement. To this end, the probability distributions of generating observations within the given finite step observation extension, after the systems generate the given observation from the two adjacent initial states, are approximate. In the future, other applications of differential privacy in DESs will be investigated.

Author Contributions: Conceptualization, Y.T. and L.Y.; methodology, Y.T.; software, Y.T.; validation, L.Y.; formal analysis, Z.L.; investigation, Y.T.; resources, Z.L.; data curation, Y.T. and L.Y.; writing—original draft preparation, Y.T.; writing—review and editing, Z.L. and N.W.; visualization, Y.T.; supervision, L.Y.; project administration, L.Y.; funding acquisition, L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Zhuhai Industry-University-Research Project with Hongkong and Macao under Grant ZH22017002210014PWC and the Science Technology Development Fund, MSAR, under Grant No. 0101/2022/A.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Abbreviations

The following abbreviations are used in this manuscript:

DES	Discrete Event System
DFA	Deterministic Finite Automaton

References

1. Oneto, L.; Fumeo, E.; Clerico, G.; Canepa, R.; Papa, F.; Dambra, C.; Mazzino, N.; Anguita, D. Dynamic delay predictions for large-scale railway networks: Deep and shallow extreme learning machines tuned via thresholdout. *IEEE Trans. Syst. Man Cybern.-Syst.* **2017**, *47*, 2754–2767. [CrossRef]
2. Xiong, J.B.; Ma, R.; Chen, L.;Tian, Y.L.; Li, Q.; Liu, X.M.; Yao, Z.Q. A personalized privacy protection framework for mobile crowdsensing in IIoT. *IEEE Trans. Ind. Inform.* **2020**, *16*, 4231–4241. [CrossRef]
3. De Prisco, R.; De Santis, A. On the relation of random grid and deterministic visual cryptography. *IEEE Trans. Inf. Forensic Secur.* **2014**, *9*, 653–665. [CrossRef]
4. Beunardeau, M.; Connolly, A.; Geraud, R.; Naccache, D. White-box cryptography: Security in an insecure environment. *IEEE Secur. Priv.* **2017**, *14*, 88–92. [CrossRef]
5. Sweeney, L. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* **2002**, *10*, 557–570. [CrossRef]
6. Zhang, B.B.; Lin, J.C.W.; Liu, Q.K.; Fournier-Vigers, P.; Djenouri, Y. A (k, p)-anonymity framework to sanitize transactional database with personalized sensitivity. *J. Internet Technol.* **2019**, *20*, 801–808. [CrossRef]
7. Lin, J.C.W.; Liu, Q.K.; Fournier-Viger, P.; Hong, T.P. PTA: An efficient system for transaction database anonymization. *IEEE Access* **2016**, *4*, 6467–6479. [CrossRef]

8. Dwork, C. Differential privacy. In Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, Venice, Italy, 10–14 July 2006; Volume 4052, pp. 1–12.
9. Dwork, C. Differential privacy: A survey of results. In Proceedings of the 5th International Conference on Theory and Applications of Models of Computation, Xi'an, China, 25–29 April 2008; Volume 4978, pp. 1–19. [CrossRef]
10. Wu, X.; Zhang, Y.T.; Shi, M.Y.; Li, P.; Li, R.R.; Xiong, N.N. An adaptive federated learning scheme with differential privacy preserving. *Future Gener. Comp. Syst.* **2021**, *127*, 362–372. [CrossRef]
11. Zhao, Y.; Chen, J.J. A Survey on differential privacy for unstructured data content. *ACM Comput. Surv.* **2022**, *54*, 3490237. [CrossRef]
12. McSherry, F.; Talwar, K. Mechanism design via differential privacy. In Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, Providence, RI, USA, 21–23 October 2007; pp. 94–103. [CrossRef]
13. Geng, Q.; Viswanath, P. The optimal noise-adding mechanism in differential privacy. *IEEE Trans. Inf. Theory* **2016**, *62*, 925–951. [CrossRef]
14. Li, C.; Miklau, G.; Hay, M.; McGregor, A.; Rastogi, V. The matrix mechanism: Optimizing linear counting queries under differential privacy. *VLDB J.* **2015**, *24*, 757–781. [CrossRef]
15. Dwork, C.; Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.* **2013**, *9*, 211–406. [CrossRef]
16. You, D.; Wang, S.G.; Zhou, M.C.; Seatzu, C. Supervisory control of Petri nets in the presence of replacement attacks. *IEEE Trans. Autom. Control* **2022**, *67*, 1466–1473. [CrossRef]
17. Saboori, A.; Hadjicostis, C.N. Notions of security and opacity in discrete event systems. In Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, LA, USA, 12–14 December 2007; pp. 3101–3106.
18. Bryans, J.W.; Koutny, M.; Ryan, P.Y.A. Modelling opacity using Petri nets. *Electron. Notes Theor. Comput. Sci.* **2005**, *121*, 101–115. [CrossRef]
19. Jones, A.; Leahy, K.; Hale, M. Towards differential privacy for symbolic systems. In Proceedings of the American Control Conference (ACC), Philadelphia, PA, USA, 10–12 July 2019; pp. 372–377.
20. Chen, B.; Leahy, K.; Jones, A.; Hale, M. Differential privacy for symbolic systems with application to Markov chains. *arXiv* **2022**, arXiv:2202.03325.
21. Tong, Y.; Li, Z.W.; Seatzu, C.; Giua, A. Verification of initial-state opacity in Petri nets. In Proceedings of the 54th IEEE Conference on Decision and Control (CDC), Osaka, Japan, 15–18 December 2015; pp. 344–349.
22. Tong, Y.; Li, Z.W.; Seatzu, C.; Giua, A. Verification of state-based opacity using Petri nets. *IEEE Trans. Autom. Control* **2017**, *62*, 2823–2837. [CrossRef]
23. Zhang, B.; Shu, S.L.; Lin, F. Maximum information release while ensuring opacity in discrete event systems. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 1067–1079. [CrossRef]
24. Ji, Y.D.; Yin, X.; Lafortune, S. Opacity enforcement using nondeterministic publicly known edit functions. *IEEE Trans. Autom. Control* **2019**, *64*, 4369–4376. [CrossRef]
25. Yang, J.N.; Cao, Y.Z.; Wang, H.P. Differential privacy in probabilistic systems. *Inf. Comput.* **2017**, *254*, 84–104. [CrossRef]
26. Hou, Y.F.; Shen, Y.N.; Li, Q.D.; Ji, Y.F.; Li, W. Modeling and optimal supervisory control of networked discrete-event systems and their application in traffic management. *Mathematics* **2023**, *11*, 10003. [CrossRef]
27. Rezig, S.; Ezzeddine, W.; Turki, S.; Rezg, N. Mathematical model for production plan optimization-a case study of discrete event systems. *Mathematics* **2020**, *8*, 60955. [CrossRef]
28. Rouabah, Y.; Li, Z.W. The unfolding: Origins, techniques, and applications within discrete event systems. *Mathematics* **2023**, *11*, 10047. [CrossRef]
29. Kumar, R.; Garg, V.K. Control of stochastic discrete event systems: Synthesis. In Proceedings of the 37th IEEE Conference on Decision and Control, Tampa, FL, USA, 18 December 1998; pp. 3299–3304.
30. Huang, Y.S.; Chiang, H.S.; Jeng, M. Fault measure of discrete event systems using probabilistic timed automata. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC), Anchorage, AK, USA, 9–12 October 2011; pp. 1218–1223.