mathematics

MDPI

Article

Verification and Enforcement of (ϵ, ξ)-Differential Privacy over Finite Steps in Discrete Event Systems

Tareq Ahmad Al-Sarayrah [1], Zhiwu Li [2,*], Guanghui Zhu [3], Mohammed A. El-Meligy [4] and Mohamed Sharaf [4]

[1] School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China
[2] Institute of Systems Engineering, Macau University of Science and Technology, Taipa 999078, Macau SAR, China
[3] School of Electrical and Mechanical Engineering, Xuchang University, Xuchang 461000, China
[4] Industrial Engineering Department, College of Engineering, King Saud University, P.O. Box 800, Riyadh 11421, Saudi Arabia
* Correspondence: zwli@must.edu.mo

Abstract: In the realm of data protection strategies, differential privacy ensures that unauthorized entities cannot reconstruct original data from system outputs. This study explores discrete event systems, specifically through probabilistic automata. Central is the protection of state data, particularly the initial state privacy of multiple starting states. We introduce an evaluation criterion to safeguard initial states. Using advanced algorithms, the proposed method counters the probabilistic identification of any state within this collection by adversaries from observed data points. The efficacy is confirmed when the probability distributions of data observations tied to these states converge. If a system's architecture does not meet state differential privacy demands, we propose an enhanced supervisory control mechanism. This control upholds state differential privacy across all initial states, maintaining operational flexibility within the probabilistic automaton framework. Concluding, a numerical analysis validates the approach's strength in probabilistic automata and discrete event systems.

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

MSC: 93C83

Citation: Al-Sarayrah, T.A.; Li, Z.; Zhu, G.; El-Meligy, M.A.; Sharaf, M. Verification and Enforcement of (ϵ, ξ)-Differential Privacy over Finite Steps in Discrete Event Systems. *Mathematics* **2023**, *11*, 4991. https://doi.org/10.3390/math11244991

Academic Editor: Qingshan Jiang

Received: 7 November 2023
Revised: 14 December 2023
Accepted: 15 December 2023
Published: 18 December 2023

1. Introduction

In the context of expansive data analytics and the handling of large, sensitive datasets, the RibsNet architecture [1] presents a significant advancement in data center networks, enhancing the management and confidentiality of extensive data collections. This innovation aligns with the ongoing efforts in data privacy, where methodologies like nearest similarity-based clustering [2], k-anonymity [3], and augmented l-diversity [4] have been developed, and both the scholarly community and the industrial sphere are diligently strengthening the confidentiality of sensitive elements within extensive databases. These advanced methods strive to balance the essential benefits of big data analytics with the ethical need for individual privacy [2]. Especially in domains of heightened sensitivity, such as healthcare, these mechanisms for preserving privacy serve as a sine qua non for engendering data-driven insights while mitigating the inherent risks posed by the disclosure of personalized data [3,4].

In data privacy, advanced anonymity architectures are being carefully improved to protect sensitive information in transactional databases. These pioneering algorithms, frequently modulated by bespoke sensitivity indices dictated by the end users, epitomize a finely calibrated balance between data obfuscation and utility. Surpassing extant paradigms in terms of computational frugality and data preservation, these algorithms prioritize

the diminution of information attrition [5]. In synchrony with the emergent paradigms in the data privacy landscape, bio-inspired computational strategies such as the black hole algorithm are being judiciously employed to surmount the conventional limitations endemic to k-anonymity models. Preliminary empirical evidence substantiates the black hole algorithm's superior capacity for amplifying data utility when juxtaposing against existing frameworks [6].

Differential privacy is a game-changing concept that precisely measures the privacy risks associated with using statistical databases. Transcending classical paradigms, this framework contends with privacy infirmities that affect even those individuals who are conspicuously absent from the database's annals. Through the deployment of a sophisticated suite of algorithms, differential privacy navigates an impeccable equilibrium between maximizing data utility and furnishing unassailable privacy guarantees. As a result, it carves out a novel ambit in the realm of secure, yet perspicacious, data analytics [7,8].

The incorporation of stochastic noise has solidified its role as a quintessential instrument for attaining differential privacy, providing a rigorous framework for individual data safeguarding within collective data assemblages. Employing exacting mathematical formulations such as ϵ-differential [9] and (ϵ, δ)-differential privacy [10], this approach critically evaluates the performance metrics of assorted noise-introduction protocols. It furnishes a nuanced exegesis of the intrinsic trade-offs between data concealment and utility, notably in the realm of consensus algorithms.

Advanced frameworks that integrate noise have been developed to protect the subtle differences between similar datasets using differential privacy principles. These frameworks facilitate the injection of stochastic noise, which adheres to various probabilistic distributions, such as the Laplacian or Gaussian models, into the results of data queries. By doing so, they fulfill the stipulated privacy assurances [10,11].

In the evolving field of differential privacy, especially for qualitative data, moving from global sensitivity models to localized sensitivity approaches is a significant change. While the exponential mechanism [12] has long served as the bedrock for global sensitivity-centric strategies, the recently unveiled local dampening mechanism [13] inaugurates a more sophisticated and malleable architecture for safeguarding data privacy. This innovative development exploits local sensitivity metrics to furnish more contextually nuanced and granular privacy assurances, thereby pioneering new frontiers in the intricate realm of qualitative data protection.

Discrete event systems (DESs) function as a formidable paradigm for the conceptualization, scrutiny, and governance of a diverse gamut of systems, wherein the conceptual entity of an 'event' is pivotal in dictating systemic dynamics. Ranging from industrial manufacturing sequences and computational networks to telecommunication infrastructures and operations research, the scope of DESs effortlessly straddles conventional disciplinary demarcations. Within this multifaceted domain, a salient focal point comprises the assessment and amelioration of security susceptibilities, especially in scenarios characterized by incomplete observability [14].

Opacity [15] stands as an indispensable facet of security within the ambit of DESs, concentrating on the extent to which a system discloses its internal confidentialities to an extraneous, non-active observer-frequently denominated as either the intruder or malevolent scrutineer. In this intellectual landscape, the notion of language-based opacity [16] takes on considerable weight. This concept delves into the inherent restrictions that come with an outsider's incomplete surveillance of a system. The critical inquiry here is whether these observational limitations inhibit the outsider from discerning whether the unfolding sequence of events betrays confidential or delicate matters.

State-based opacity [17] augments the analytical tableau by imbuing it with discrete temporal dimensions that enrich the overarching conceptual framework. Contingent upon the particular temporal instances at which a system transits through confidential states, this genre of opacity can be meticulously categorized into four seminal subclasses: initial-state opacity [18], current-state opacity [19], k-step opacity [20], and infinite-step

opacity [21]. Each of these refined categories furnishes a stratified comprehension of a system's robustness vis-à-vis unauthorized external examination across diverse temporal vicissitudes. Notwithstanding its pivotal significance, the authentication of opacity within the frameworks of DESs is anything but elementary, frequently confronting formidable computational impediments. Cutting-edge inquiries in the discipline delve into the architectural constituents of automata paradigms that could potentially facilitate the more manageable verification of opacity [22].

Within the intricate nexus of differential privacy and symbolic control architectures, the transposition of statistical privacy schemata to non-quantitative datasets establishes a pioneering standard. Leveraging exponential methodologies and specialized automata configurations, sensitive alphanumeric sequences are skillfully approximated. This is executed with a dual objective: the preservation of informational sanctity and the retention of data utility. The governance of this intricate process is underscored by the employment of distance metrics such as the Levenshtein distance [23].

At this interdisciplinary juncture, the enhancement of data security is amplified across a diverse spectrum of fields, ranging from natural language processing to intricate industrial systems. Predicated upon this foundational understanding, the current scholarly endeavor aims to further elevate this paradigm. The focus is sharpened on safeguarding the opacity of initial states within DESs, utilizing probabilistic automata embedded within the stringent confines of differential privacy. Within the realm of discrete-event systems, the notion of initial state opacity emerges as an indispensable yardstick for assessing the system's prowess in concealing its nascent state from unauthorized external entities.

Classical deterministic frameworks for initial state opacity have developed to encompass probabilistic and stochastic paradigms, thus facilitating a more textured comprehension of security dynamics in volatile environments. To cater to elevated privacy requisites, especially in situations where an intrusive entity possesses comprehensive structural awareness of the system, advanced iterations of opacity, such as robust initial state opacity, have been formulated [19,24,25].

Techniques for substantiating these robust opacity conditions are undergoing refinement via innovative approaches, including parallel composition techniques and integer linear programming algorithms [26]. Initial state opacity is a crucial area that combines verification methods, probability models, and cybersecurity concepts to effectively assess data protection in dynamic systems. Cutting-edge computational schemas, such as Petri net models, facilitate the expeditious verification of initial state opacity, obviating the necessity for laborious state space enumeration. Contemporary real-time validation methodologies, encompassing linear programming algorithms, further enhance the relevance and applicability of initial state opacity in intricate network architectures, including those inherent to defense systems and mobile communications networks [27,28].

In architectures delineated via non-deterministic finite automata and their probabilistic counterparts, i.e., probabilistic finite automata, the verification of initial state opacity necessitates sophisticated computational modalities and analytical techniques. The complexity is exacerbated when extended to non-deterministic transition systems, particularly in the instances where state spaces are potentially unbounded. The exploration of initial state opacity thus constitutes a critical nexus between formal computational methodologies, automata theory, and information security, engendering both algorithmic intricacies and theoretical conundrums [29,30]. Differential privacy was delicately integrated into discrete event systems using probabilistic automata in a vital piece of research [31]. The major goal was to protect state information illustrating system resource settings. This technique was designed to provide state differential privacy, with an emphasis on thorough, step-by-step validation. This method made it difficult for potential adversaries to infer the system's starting state from a limited collection of observations.

In an era where the landscape of extensive data analysis is rapidly evolving, the necessity for impregnable data privacy frameworks becomes ever more apparent. The research presented herein constitutes a substantial breakthrough in the sphere of differential

privacy within the context of DESs. This study introduces the concept of the privacy decay factor [32], denoted as ξ, a groundbreaking development that diverges markedly from preceding research. Prior studies primarily concentrated on the protection of state information within DESs through the utilization of probabilistic automata. This research, however, forges new paths by integrating this innovative parameter. The implementation of ξ has proven to be particularly efficacious in ensuring the privacy of initial states or conditions, thereby guaranteeing the secure concealment of sensitive data from the very outset of data processing a pivotal requirement in the face of continually evolving privacy standards.

By broadening the scope to encompass an expanded range of initial state conditions and databases, this approach markedly bolsters the method's resilience and adaptability. The incorporation of ξ addresses a significant void in existing methodologies, providing a more fortified framework for preserving the privacy of initial states against potential adversaries. This methodical consideration of an extensive array of starting conditions not only augments the theoretical foundations of differential privacy within DESs but also showcases its tangible applicability across a spectrum of intricate systems.

Building on this essential understanding, the present scholarly pursuit endeavors to advance this paradigm further. It concentrates on the fortification of the privacy of initial states within DESs, employing probabilistic automata encased within the stringent parameters of differential privacy. As a consequence, this research establishes a new benchmark in the domain of privacy preservation, signaling a shift in data security strategies across diverse sectors. This is particularly pertinent in industries where stringent privacy measures from the commencement of data processing are of the utmost importance. The following lines describe the main contributions of this scientific endeavor:

- The research introduces a novel model of (ϵ, ξ)-differential privacy for discrete event systems (DESs) formulated by probabilistic automata, ensuring equitable resource distribution across multiple initial states.
- A novel verification strategy is introduced, originating from distinct observations of a vast set of initial states and evaluating adherence to the tenets of (ϵ, ξ)-differential privacy across defined observational sequences.
- By seamlessly integrating probabilistic automata with a diverse set of initial states, the study presents a tailored verification approach. Should systems deviate from the privacy standards, a specialized control mechanism is deployed, conveying (ϵ, ξ)-differential privacy within the overarching closed-loop system.
- The research's potency is exemplified through a detailed numerical case study, illustrating the verification method's acumen in assessing the privacy integrity of specific automata classes.

The rest of this paper is organized in a way that facilitates a comprehensive study of the relevant topics. Section 2 sets the foundation by introducing the basics of probabilistic automata and the important principles of (ϵ, ξ)-differential privacy in the context of data security. Section 3 serves as the main investigative focus of the study. The approach to the verification of (ϵ, ξ)-differential privacy over finite steps is detailed in Section 4. In Section 5, we describe a method for ensuring (ϵ, ξ)-differential privacy via supervisory control. A numerical case study is presented in Section 6 to offer empirical support for the methodologies proposed. Finally, Section 7 concludes the paper, summarizing the key findings and their implications.

2. Preliminaries

This section introduces the concept of probabilistic automata within the framework of discrete event systems and discusses the conventional notion of (ϵ, ξ)-differential privacy.

2.1. Probabilistic Automata in Discrete Event Systems (DESs)

In the study of DESs, the use of deterministic finite automata is crucial. A deterministic finite automaton can be formally described by a structure $\mathcal{D} = (S, \Sigma, \delta, s_0)$ [33], where S

is a finite set of states, and Σ is the set of events, which can be further divided into Σ_o for the set of observable events and Σ_{uo} for that of unobservable events. The partial transition function $\delta : S \times \Sigma \to S$ defines the deterministic behavior of the DES, mapping a state and an event to a subsequent state. Finally, s_0 denotes the initial state, taken from the set S, before any events have occurred.

To grasp the operation of the deterministic finite automaton, consider $\delta(s, e)$ to be defined if event e from the set Σ can trigger a transition from state s. Adapting δ for event sequences or strings, we have $\delta : S \times \Sigma^* \to S$. This function operates as per $\delta(s, \varepsilon) = s$ and $\delta(s, ue) = \delta(\delta(s, u), e)$, given that both $\delta(s, u)$ and $\delta(\delta(s, u), e)$ are defined for a state s and a string combination u. For any state s and a string u, if δ is appropriately defined for the string u at state s, it is denoted as $\delta(s, u)!$. The length of a string u, indicating the number of events in it, is denoted by $|u|$.

For an automaton $\mathcal{D} = (S, \Sigma, \delta, s_0)$ and a specific state s, the language that the automaton generates, starting from state s, is given by $L(\mathcal{D}, s) = \{u \in \Sigma^* | \delta(s, u)!\}$. In scenarios in which potential attackers can observe and log only the observable events, a key tool to consider is the natural projection, denoted by $P : \Sigma^* \to \Sigma_o^*$. This projection effectively translates any executed string within the system into a corresponding sequence of observable events, termed an observation. The definition of this projection is recursive: for any string u from Σ^* and an event e from Σ, the projection is defined as $P(ue) = P(u)P(e)$. It is important to note that $P(e)$ is equal to e when e belongs to Σ_o, and it becomes the empty string ε when e belongs to Σ_{uo}. Building on this, the set of observations generated by an automaton $\mathcal{D} = (S, \Sigma, \delta, s_0)$ starting from a state s in S can be defined as

$$L_o(\mathcal{D}, s) = \{\omega \in \Sigma_o^* | \exists u \in L(\mathcal{D}, s) : P(u) = \omega\}.$$

This essentially captures all observable sequences corresponding to the possible behaviors of the automaton from the state s.

Definition 1 (Probabilistic automaton [34])**.** *A probabilistic automaton is defined as a tuple* $\mathcal{G} = (\mathcal{D}, \rho)$*, where*

- $\mathcal{D} = (S, \Sigma, \delta, s_0)$ *denotes the underlying deterministic finite automaton,*
- $\rho : S \times \Sigma \to [0, 1]$ *serves as a probability distribution function over transitions.*

Again, S *is the set of states, and* $s_0 \in S$ *is the initial state. Given a state* $s \in S$ *and an event* $e \in \Sigma$*,*

$$\rho(s, e) = \begin{cases} 0, & \text{if } \delta(s, e) \text{ is undefined} \\ > 0, & \text{if } \delta(s, e) \text{ is defined,} \end{cases}$$

the set of feasible events at a given state s *is* $E(s) = \{e \in \Sigma \mid \rho(s, e) > 0\}$*, subject to the constraint* $\sum_{e \in E(s)} \rho(s, e) = 1$*.*

When the underlying structure is implied, the probabilistic automaton $\mathcal{G}$ can be denoted succinctly as $\mathcal{G}(s_0)$. Given a probabilistic automaton $\mathcal{G} = (S, \Sigma, \delta, s_0, \rho)$, the metric Pr_σ plays a pivotal role in determining the likelihood of generating a string ue in Σ^* from state s [34]. When the string ue consists of $u \in \Sigma^*$ and an event $e \in \Sigma$, Pr_σ is recursively defined as

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

where ε within Σ^* is the empty string. At its core, $Pr_\sigma(s, u)$ signifies the probability that the string u is executed starting from state s in the automaton $\mathcal{G}$. A positive value of $Pr_\sigma(s, u)$ is realized if $\delta(s, u)$ for every $u \in \Sigma^*$. In this framework, strings and observations emerge as foundational constructs. A string, represented within Σ^*, takes the form $u : \mathbb{N} \to \Sigma$, where Σ is the finite alphabet set of events. Such a string can be visualized as a deterministic

sequence, $u = u_1 u_2 \dots u_n$, with each $u_i \in \Sigma$. The automaton's transition function establishes that the combination of a state and a string leads to a unique subsequent state, as represented by $\delta : S \times \Sigma^* \to S$. Further, the language emerging from state s is denoted by

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

highlighting all strings that can be produced from state s with a non-zero probability.

Contrasting with strings, observations are denoted as ω within Σ_o^*. They are defined as $\omega : \mathbb{N} \to \Sigma_o$, with $\Sigma_o \subseteq \Sigma$ being the observation alphabet set of events. Observations stem from a potentially non-bijective projection, formalized as $P : \Sigma^* \to \Sigma_o^*$.

This suggests that multiple strings might map to the same observation. Essentially, observations capture a more abstract or condensed perspective of the automaton's behavior. The observations generated from a state s are defined as

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

This definition describes all possible observations inferred from the strings originating at state s. To further delineate the relationship between strings and observations, we can define the set of strings that are consistent with a particular observation when the automaton is in state s, which is given by

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

The probability of generating a specific observation from a state s is calculated as

$$Pr(s, \omega) = \sum_{u \in U(s,\omega)} Pr_\sigma(s, u).$$

While strings u outline the explicit sequences of transitions an automaton undergoes, observations ω act as potential aggregated or abstracted representations of these sequences. These mathematical representations and definitions bind the automaton's dynamics, bridging the states to the strings and the resultant observations.

Example 1. *Consider the probabilistic automaton structure $\mathcal{G} = (S, \Sigma, \delta, \rho)$ illustrated in Figure 1. The automaton initiates from the state s_0, leading us to denote the automaton configured in this manner as $\mathcal{G}(s_0)$. This configuration comprises a set of states $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$ and partitions the event set Σ into observable events $\Sigma_o = \{\alpha, \beta, \lambda, \gamma, \mu\}$ and unobservable events $\Sigma_{uo} = \{\tau\}$. For this configuration, $\rho(s_0, \alpha) = 0.8$ and $\rho(s_0, \lambda) = 0.2$. Moreover, we ascertain that $\sum_{e \in E(s_0)} \rho(s_0, e) = 1$ with $E(s_0) = \{\alpha, \lambda\}$. Considering a string $u = \alpha\gamma\tau$, the probability of generating the string u from s_0 is given by $\Pr_\sigma(s_0, \alpha\gamma\tau) = \rho(s_0, \alpha) \times \rho(s_1, \gamma) \times \rho(s_2, \tau) = 0.24$. However, let $\omega = \alpha\gamma$. Then, the probability of generating this sequence of observation is given by $\Pr(s_0, \omega) = \Pr_\sigma(s_0, \alpha\gamma) + \Pr_\sigma(s_0, \alpha\gamma\tau) = 0.84$.*

Given an automaton $\mathcal{G}$, the set of states reached by generating a string consistent with observation ω from state s is

$$\phi(s, \omega) = \{s' \in S | \exists u \in U(s, \omega) : \delta(s, u) = s'\}.$$

The probability of transitioning from state s_0 to state s given observation ω is denoted as $Pr(s|\phi(s_0, \omega))$. For a state s' that is reachable after generating an observation ω from the initial state s_0, the probability of generating a string u is given by $Pr_\sigma(s_0, \omega, s, u) = Pr(s|\phi(s_0, \omega)) \times Pr_\sigma(s, u)$. For the purpose of elucidating the mathematical constructs utilized in our analysis, we define an indicator function, denoted as I, which ascertains the validity of state transitions. Formally, it is defined as

$$I_{\delta(s_0,u)=s} = \begin{cases} 1, & \text{if } \delta(s_0, u) = s \\ 0, & \text{otherwise.} \end{cases}$$

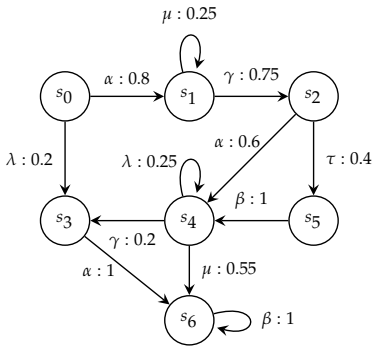

Figure 1. A probabilistic automaton $\mathcal{G}$.

In complex mathematical models, the relationship between system changes and observations is subtle. Although not immediately obvious, the system's current state becomes clear after careful analysis of these observation sequences [35]. Delving deeper into such structures, a crucial concept that becomes intertwined with our discussion is the expectation under a specific distribution. In the provided formulation, $\mathbb{E}_{u\sim U(s_0,\omega)}$ signifies the expected value (or average) taken over all strings u drawn from the set $U(s_0,\omega)$. This encapsulates the average behavior or outcome under the distribution of strings originating from s_0 and culminating in observations ω. Furthermore, by harnessing the power of the indicator function, the probability $Pr(s|\phi(s_0,\omega))$ undergoes a modification, being redefined as an expectation:

$$Pr(s|\phi(s_0,\omega)) = \frac{\mathbb{E}_{u\sim U(s_0,\omega)}[I_{\delta(s_0,u)=s} \times Pr_\sigma(s_0,u)]}{\mathbb{E}_{u\sim U(s_0,\omega)}[Pr_\sigma(s_0,u)]}.$$

This expression quantifies the likelihood that the automaton, initiating its sequence at state s_0 and transitioning with strings from the set $U(s_0,\omega)$, lands on state s. Subsequently, the probability $Pr_\sigma(s_0,\omega,s,u)$ integrates the indicator function and the expectation concept:

$$Pr_\sigma(s_0,\omega,s,u) = \begin{cases} \frac{\sum_{u\in U(s_0,\omega)} I_{\delta(s_0,u)=s} \times Pr_\sigma(s_0,u)}{\sum_{u\in U(s_0,\omega)} Pr_\sigma(s_0,u)} \times Pr_\sigma(s,u) & \text{if } s \in \phi(s_0,\omega) \\ 0 & \text{otherwise.} \end{cases}$$

This formulation captures the normalized probability of the system transitioning from s_0 to s under the observation ω and then executing string u from state s.

Example 2. *Consider the probabilistic automaton depicted in Figure 1, with the initial state s_0. Let $\Sigma_o = \{\alpha,\beta,\lambda,\gamma,\mu\}$ and $\Sigma_{uo} = \{\tau\}$. For the observation $\omega = \alpha\gamma$, define $U(s_0,\alpha\gamma)$ as the set of all strings originating from s_0 that are consistent with ω, i.e., $U(s_0,\alpha\gamma) = \{\alpha\gamma\tau,\alpha\gamma\}$. The cumulative conditional probabilities for transitions from s_0 to either state s_2 or state s_5 using expectations are:*

$$\begin{aligned} Pr(s_2|\phi(s_0,\alpha\gamma)) &= \frac{\mathbb{E}_{u\sim U(s_0,\alpha\gamma)}[I_{\delta(s_0,u)=s_2} \times Pr_\sigma(s_0,u)]}{\mathbb{E}_{u\sim U(s_0,\alpha\gamma)}[Pr_\sigma(s_0,u)]} \\ &= 0.6/(0.6+0.24) = 5/7; \\ Pr(s_5|\phi(s_0,\alpha\gamma)) &= \frac{\mathbb{E}_{u\sim U(s_0,\alpha\gamma)}[I_{\delta(s_0,u)=s_5} \times Pr_\sigma(s_0,u)]}{\mathbb{E}_{u\sim U(s_0,\alpha\gamma)}[Pr_\sigma(s_0,u)]} \\ &= 0.24/(0.6+0.24) = 2/7. \end{aligned}$$

Consider the transition scenario from the initial state s_0 to state s_2. In this instance, the probability of executing the string $u = \tau\beta\lambda$ from state s_2 is calculated as $Pr_\sigma(s_0,\alpha\gamma,s_2,\tau\beta\lambda) = 5/7 \times 0.4 \times 1 \times 0.25 = 1/14$. Conversely, considering the transition to state s_5, the probability

of executing the string $u = \beta\lambda\mu$ *from state* s_5 *is denoted as* $Pr_\sigma(s_0, \alpha\gamma, s_5, \beta\lambda\mu) = 2/7 \times 1 \times 0.25 \times 0.55 = 11/280$.

Let $\mathbb{N}$ be the set of natural numbers, and $\mathbb{N}^+$ be the set of positive natural numbers, i.e., $\mathbb{N}^+ = \{x | x > 0 \wedge x \in \mathbb{N}\}$. For an observation $\omega \in \Sigma_o^*$, the collection of all observations produced from a state $s \in \phi(s_0, \omega)$ at $k \in \mathbb{N}^+$ step is defined as [34]

$$\ell(s_0, \omega, s, k) = \{\omega' \in \Sigma_o^* | \omega' \in \ell(\mathcal{G}, s), |\omega'| = k\}.$$

The comprehensive set of all possible observations that arise from k-step observation extensions with $k \in \mathbb{N}^+$, following the generation of an observation ω from state s_0, is

$$\ell(s_0, \omega, k) = \bigcup_{s \in \phi(s_0, \omega)} \ell(s_0, \omega, s, k).$$

The likelihood of producing $\omega' \in \ell(s_0, \omega, k)$ after the system has yielded an observation ω from s_0 is defined as

$$Pr(s_0, \omega, k, \omega') = \sum_{s \in \phi(s_0, \omega)} \sum_{u \in U(s, \omega')} Pr_\sigma(s_0, \omega, s, u).$$

It should be highlighted that $Pr(s_0, \omega, k, \omega')$ denotes the probability of observing ω' after a k-step extension, assuming the prior generation of ω from s_0.

Example 3. *Consider the probabilistic automaton illustrated in Figure 1, where* s_0 *serves as the initial state. Suppose* $\Sigma_o = \{\alpha, \gamma, \lambda, \beta, \mu\}$ *and* $\Sigma_{uo} = \{\tau\}$. *Let us assume* $k = 2$. *For the observation sequence* $\omega = \alpha\gamma$ *generated from state* s_0, *we can represent the sequence of observation extensions over two steps from either state* s_2 *or* s_5 *as follows:* $\ell(s_0, \alpha\gamma, s_2, 2) = \{\alpha\gamma, \alpha\lambda, \alpha\mu\}$, *and* $\ell(s_0, \alpha\gamma, s_5, 2) = \{\beta\mu, \beta\gamma, \beta\lambda\}$. *Combining these sequences, we find the set of sequences that lead to either* s_2 *or* s_5 *over two steps after generating the observation sequence* $\omega = \alpha\gamma$ *from state* s_0 *are* $\ell(s_0, \alpha\gamma, 2) = \ell(s_0, \alpha\gamma, s_2, 2) \cup \ell(s_0, \alpha\gamma, s_5, 2) = \{\alpha\gamma, \alpha\lambda, \alpha\mu, \beta\mu, \beta\gamma, \beta\lambda\}$. *Here,* $Pr(s_0, \omega, 2, \omega')$ *is calculated for each possible sequence as follows:*

$$\begin{aligned}
&\textit{For } \omega' = \alpha\lambda: && Pr(s_0, \alpha\gamma, 2, \alpha\lambda) = 5/7 \times 0.6 \times 0.25 = 3/28;\\
&\textit{For } \omega' = \alpha\gamma: && Pr(s_0, \alpha\gamma, 2, \alpha\gamma) = 5/7 \times 0.6 \times 0.2 = 3/35;\\
&\textit{For } \omega' = \alpha\mu: && Pr(s_0, \alpha\gamma, 2, \alpha\mu) = 5/7 \times 0.6 \times 0.55 = 33/140;\\
&\textit{For } \omega' = \beta\mu: && Pr(s_0, \alpha\gamma, 2, \beta\mu) = Pr_\sigma(s_0, \alpha\gamma, s_2, \tau\beta\mu) + Pr_\sigma(s_0, \alpha\gamma, s_5, \beta\mu)\\
& && \quad = 5/7 \times 0.4 \times 1 \times 0.55 + 2/7 \times 0.55 \times 1 = 11/35;\\
&\textit{For } \omega' = \beta\lambda: && Pr(s_0, \alpha\gamma, 2, \beta\lambda) = Pr_\sigma(s_0, \alpha\gamma, s_2, \tau\beta\lambda) + Pr_\sigma(s_0, \alpha\gamma, s_5, \beta\lambda)\\
& && \quad = 5/7 \times 0.25 \times 1 \times 0.4 + 2/7 \times 0.25 \times 1 = 1/7;\\
&\textit{For } \omega' = \beta\gamma: && Pr(s_0, \alpha\gamma, 2, \beta\gamma) = Pr_\sigma(s_0, \alpha\gamma, s_2, \tau\beta\gamma) + Pr_\sigma(s_0, \alpha\gamma, s_5, \beta\gamma)\\
& && \quad = 5/7 \times 0.2 \times 1 \times 0.4 + 2/7 \times 0.2 \times 1 = 4/35.
\end{aligned}$$

2.2. Differential Privacy

Differential privacy is a framework that quantifies the privacy guarantees offered by a randomized algorithm. It provides a measure ensuring that the output of a computation remains approximately the same, even if one record in the input dataset is altered. This ensures that an adversary cannot determine whether a specific individual's information is included in the input to the function. Formally, given a threshold $\epsilon > 0$, a randomized algorithm M adheres to ϵ-differential privacy if, for any pair of datasets A_1 and A_2 that differ by at most one record, and for any output set O, the following inequality holds [36]:

$$e^{-\epsilon} \leq \frac{P(M(A_1) \in O)}{P(M(A_2) \in O)} \leq e^{\epsilon},$$

where $M(A_1)$ and $M(A_2)$ denote the outputs of algorithm M when run on datasets A_1 and A_2, respectively. The function P assigns a probability to a potential output of M. The parameter ϵ, often referred to as the privacy budget, sets a limit on the allowable information leakage, where $\epsilon \in \mathbb{R}$ and ϵ is a non-negative real number.

In the field of differential privacy, one of the central challenges emerges from the recurrent utilization of a privacy-preserving mechanism. As this mechanism is used repeatedly, the cumulative privacy guarantees can degrade, potentially weakening the overall privacy assurance [32]. Addressing this issue necessitates a more refined strategy, and introducing a decay factor serves this purpose. This factor places emphasis on the time-dependent modification of privacy loss parameters. Consider a scenario where mechanism M is invoked in an iterative fashion. A direct or naïve summation of the privacy loss parameter ϵ over each use could swiftly consume the allotted privacy budget, compromising the robustness of the privacy safeguards in place. This predicament underscores the relevance of the decay factor. When expressed as

$$\epsilon' = \epsilon e^{-\xi i},$$

it signifies an exponential decay in the privacy loss parameter. This ensures that with each subsequent application or iteration, there is a diminishing allowance for deviation in privacy guarantees, thereby systematically curtailing potential privacy breaches. From a mathematical perspective, the compounded degradation in differential privacy over n iterations can be encapsulated by: $\epsilon_{\text{total}} = \sum_{i=1}^{n} \epsilon e^{-\xi i}$. This culminates in the refined differential privacy relationship:

$$e^{-\epsilon_{\text{total}}} \leq \frac{P(M(A_i) \in O)}{P(M(B_i) \in O)} \leq e^{\epsilon_{\text{total}}}.$$

This integration suggests that the privacy safeguard, when enhanced with a decay factor, does not merely sum over multiple invocations. Instead, it decays, emphasizing a gradual tightening of privacy constraints. This sophisticated model ensures that our differential privacy implementations remain robust and effective, even with repeated applications. While ϵ_{total} is crucial in a broad range of differential privacy applications, the study in this paper specifically emphasizes the application of differential privacy to automata initial states over finite steps. We deliberately choose to focus on the decay of ϵ across iterations and, consequently, omit the consideration of ϵ_{total} to maintain a concise and focused presentation.

Definition 2 (Decay factor ξ)**.** *A decay factor ξ in the context of differential privacy is a measure that determines the rate at which the privacy loss parameter ϵ decreases for each iteration of a privacy-preserving algorithm. It quantifies the exponential reduction in the potential privacy loss, thus ensuring enhanced long-term protection of privacy in iterative processes.*

The role of ξ includes (1) exponential decay—ξ results in an exponential decrease in ϵ' with each iteration; (2) privacy degradation control—adjusting ξ fine-tunes the privacy degradation rate, with a higher ξ enhancing protection; (3) robustness in repeated use—ξ limits cumulative privacy loss, ensuring robustness in differential privacy mechanisms over multiple iterations.

3. Investigative Focus

This section delves into the complexity of maintaining (ϵ, ξ)-differential privacy within an ensemble of probabilistic automata with multiple initial states. Two main situations are examined: the first deals with the verification of privacy parameters across automata, and the second presents an oversight framework to assure compliance operations under defined privacy restrictions.

3.1. Proximate States and Differential Privacy

Definition 3. *(Proximate states for n initials). Given a probabilistic automaton structure $\mathcal{G} = (S, \Sigma, \delta, \rho)$ and a set of initial states $\{s_0^1, s_0^2, \ldots, s_0^n\} \subseteq S$, the states in this set are said to be*

collectively proximate if there exists an observation $\omega \in \Sigma_o^* \setminus \{\varepsilon\}$ *such that for any* s_0^i *with* $1 \leq i \leq n$, $Pr(s_0^i, \omega) > 0$. *The collective contiguity implies that an observation that is not the empty string can be generated from every initial state in the set.*

Definition 4 (*K-step* (ϵ, ξ)*-differential privacy for n initials*)**.** *Let* $\mathcal{G} = (S, \Sigma, \delta, \rho)$ *denote a probabilistic automaton structure. Given a set of collectively proximate initial states* $\{s_0^1, s_0^2, \ldots, s_0^n\} \subseteq S$, *this leads to n distinct probabilistic automata* $\mathcal{G}(s_0^1), \mathcal{G}(s_0^2), \ldots, \mathcal{G}(s_0^n)$. *These automata are said to uphold* (ϵ, ξ)*-differential privacy over a k-step observation extension, modulated by a decay factor* ξ, *if for each automaton* $\mathcal{G}(s_0^i)$ *and any observation* $\omega \in \Sigma_o^*$ *originating from* s_0^i, *there exists a set of subsequent observations* $\Omega_{k'}$ *such that for each* $k' \leq k$, $\Omega_{k'} = \bigcup_{i=1}^{n} \ell(s_0^i, \omega, k')$, *where* ℓ *denotes an observation likelihood function.*

Additionally, for any two distinct initial states s_0^i and $s_0^j \in S$, and for all $k' \leq k$, given any observation $\omega' \in \Omega_{k'}$, where ω', the (ϵ, ξ)-differential privacy condition $|Pr(s_0^i, \omega, k', \omega') - Pr(s_0^j, \omega, k', \omega')| \leq \epsilon e^{-\xi k'}$ must hold. The privacy parameter ϵ undergoes exponential decay influenced by k' steps of observation, guided by the decay rate ξ. As observations accrue, the (ϵ, ξ)-differential privacy assurance wanes, underscoring the inherent challenge in preserving privacy as more data emerges.

3.2. Verification Concerns

Within an ensemble of n probabilistic automata, derived from a foundational automaton with n distinct initial states as $\{\mathcal{G}(s_0^1), \mathcal{G}(s_0^2), \ldots, \mathcal{G}(s_0^n)\}$, the verification process is paramount. It is essential to ascertain that each automaton in the ensemble genuinely observes the principles of (ϵ, ξ)-differential privacy over a k-step observation extension, particularly when influenced by a decay factor ξ.

A dedicated verifier, V_ω^k, is posited to play this crucial role. Its function is to assess whether each automaton in the ensemble, when initialized from its respective initial state within the set $\{s_0^1, s_0^2, \ldots, s_0^n\}$, indeed adheres to the (ϵ, ξ)-differential privacy criteria. More specifically, given any observation ω that originates from these initial states, following the definition of proximate states, V_ω^k must validate:

- The emergence and integrity of subsequent observations $\Omega_{k'}$ based on the defined observation likelihood function ℓ.
- The preservation of the (ϵ, ξ)-differential privacy condition between any two distinct initial states for all possible $k' \leq k$ steps of observation.

This verification serves as a continuous safeguard, ensuring that the (ϵ, ξ)-differential privacy guarantees, despite the probabilistic behaviors and intertwined initial states, remain uncompromised and robust.

4. Verification of (ϵ, ξ)-Differential Privacy over Finite Steps

For the intricate analysis of probabilistic automaton with multiple initial states, the verifier emerges as a pivotal tool. It meticulously dissects each initial state, treating it as the epicenter of an independent probabilistic automaton system. Through this isolated examination, the verifier offers a holistic understanding of every conceivable behavior that might emanate from a particular initial state. In essence, by simulating each initial state coupled with its observation sequence as a stand-alone probabilistic automaton, the verifier elucidates the potential trajectory of the entire automaton. This methodology, delineated in Algorithm 1, crafts a refined lens for researchers to discern the nuances of behaviors within a multi-initial-state probabilistic automaton framework.

Consider a set of n probabilistic automata, denoted as $\mathcal{G} = [\mathcal{G}(s_0^1), \mathcal{G}(s_0^2), \ldots, \mathcal{G}(s_0^n)]$, with each automaton defined by $\mathcal{G}(s_0^i) = (S, \Sigma, \delta, s_0^i, \rho)$. The goal of the algorithm is to compute a *Verifier* for an observation ω, denoted as $V_\omega^k = (S_v, \Sigma_o, \delta_v, S_0)$. In the heart of our algorithm lies the concept of memoization, a ubiquitous technique in algorithmic optimization. We introduce a memoization function, defined as $M : (S \times \Sigma) \to 2^S$, where the domain of M encapsulates the Cartesian product of states and events, and its codomain

is the power set of S, embracing all conceivable subsets of reachable states. Formally articulated, for any state-event pair (s, e), we have

$$M(s, e) = \{s' \in S : \delta(s, e) = s'\}.$$

At the inception of the algorithm, M is initialized as an empty function. As the algorithm progresses and each state-event pair is encountered, the resulting set of reachable states is cached in M to prevent redundant computations in future iterations. With the memoization table M established, the algorithm begins by iterating through each initial state s_0^i of $\mathcal{G}$. Depending on the current segment of the observation sequence and the state under consideration, the set of reachable states is either computed afresh using the transition function δ or promptly retrieved from M. As the algorithm traverses the observation sequence, it accumulates the resulting reachable states into S_0. Subsequently, the algorithm embarks on computing the product states, represented as $Q_1 \times Q_2 \times \ldots \times Q_n$. These product states are deduced based on observable events and the current state, leading to the formation of S_v, which is synthesized from the collection Sr. The worst-case approximated complexity of Algorithm 1 is $O\big(2^n \times n^2 \times |\Sigma_o| \times |\Sigma_{uo}|\big)$.

Definition 5 (*Verifier*). *Given a probabilistic automaton structure $\mathcal{G} = (S, \Sigma, \delta, \rho)$, and a list of proximate initial states $[s_0^1, s_0^2, \ldots, s_0^n]$ leading to n distinct lists of probabilistic automata $[\mathcal{G}(s_0^1), \mathcal{G}(s_0^2), \ldots, \mathcal{G}(s_0^n)]$, a differentially private verifier V_ω^k for a k-step observation extension modulated by a decay factor ξ is defined as a 4-tuple $V_\omega^k = (S_v, \Sigma_o, \delta_v, S_0)$. Here, S_v is the cross-product of states in the automata set, $S = Q_1 \times Q_2 \times \ldots \times Q_n$, with each Q_i being a subset of states from $\mathcal{G}(s_0^i)$, Σ_o denotes all observable events, $\delta_v : S_v \times \Sigma_o \to 2^{S_v}$ represents the transition function of the verifier, and S_0 is redefined as the Cartesian product of sets obtained from the function ϕ applied to each initial state and the observation ω, specifically $S_0 = \phi(s_0^1, \omega) \times \ldots \times \phi(s_0^n, \omega)$. This verifier V_ω^k encapsulates the collective behaviors of the original probabilistic automata set, over a k-step of observation.*

Theorem 1. *Given a verifier V_ω^k that integrates the state-transition model with respect to observation ω up to step k, an initial positive parameter ϵ, a decay factor ξ, and a list of n initial states $[s_0^1, s_0^2, \ldots, s_0^n]$, Algorithm 2 will ascertain whether the system upholds (ϵ, ξ)-differential privacy across k steps within the modulating privacy parameter ϵ that evolves with each step.*

Proof. Initiating with the assignment of a unit probability to each originating state s at $t = 0$, the procedure warrants unequivocal certainty for these foundational states. The algorithm then invokes Algorithm 1 to procure the transition probabilities $\phi(s_0^i, \omega\omega')$, thereby forming a probabilistic mapping from any state s_0^i to another s_0^j contingent on the observation sequence ω. Iteratively, the algorithm explores the subsequent potential states for each event in the observation sequence, thereby crafting a dynamic probabilistic landscape for each state s.

Central to the verification of privacy is the computation of the absolute deviation between the transition probabilities of any two initial states s_0^i and s_0^j. If any such deviation surpasses ϵ, the algorithm returns a negative result. As the observation progresses, the state set S undergoes updates to encompass new feasible states, symbolized as S^*. Should the algorithm traverse all k steps without infringing the evolving (ϵ, ξ)-differential privacy constraint, it returns an affirmative outcome. This, by induction, validates the algorithm's fidelity to (ϵ, ξ)-differential privacy for all inaugural states across steps $1 \leq k' \leq k$, culminating in the affirmation of the theorem. □

Algorithm 1: Construction of a k-step verifier

Input : A list of n probabilistic automata $\mathcal{G} = [\mathcal{G}(s_0^1), \ldots, \mathcal{G}(s_0^n)]$, and observation ω of length k
Output: A verifier $V_\omega^k = (S_v, \Sigma_o, \delta_v, S_0)$

$M \leftarrow \emptyset$; $Sr \leftarrow \emptyset$; $St \leftarrow \emptyset$
foreach s *in* $[s_0^1, \ldots, s_0^n]$ **do**
  $M[s, \omega[1]] \leftarrow \emptyset$; $e \leftarrow \omega[1]$; $\phi(s,e) \leftarrow \emptyset$; $Ws \leftarrow \{s\}$
  foreach s' *in* Ws **do**
    $Ws \leftarrow Ws \setminus \{s'\}$
    foreach e' *in* $\{e\} \cup \Sigma_{uo}$ **do**
      if $\delta(s', e') = s''$ **then**
        $\phi(s,e) \leftarrow \phi(s,e) \cup \{s''\}$ `// If` $e' = e$
        $Ws \leftarrow Ws \cup \{s''\}$ `// If` $e' \in \Sigma_{uo}$
$S_0 \leftarrow \phi(s_0^1, \omega) \times \ldots \times \phi(s_0^n, \omega)$
foreach s *in* $[s_0^1, \ldots, s_0^n]$ **do**
  for $i = 2$ **to** $|\omega|$ **do**
    $e \leftarrow \omega[i]$; $\omega' \leftarrow \omega[1] \ldots \omega[i-1]$; $\omega'' \leftarrow \omega[i]$; $\phi(s, \omega'') \leftarrow \emptyset$
    foreach s' *in* $\phi(s, \omega')$ **do**
      if $M[s', e] \neq \emptyset$ **then**
        $\phi(s', e) \leftarrow M[s', e]$
      else
        $Ws \leftarrow \{s'\}$
        foreach s'' *in* Ws **do**
          $Ws \leftarrow Ws \setminus \{s''\}$
          foreach e' *in* $\{e\} \cup \Sigma_{uo}$ **do**
            if $\delta(s'', e') = s'''$ **then**
              $\phi(s', e) \leftarrow \phi(s', e) \cup \{s'''\}$ `// If` $e' = e$
              $Ws \leftarrow Ws \cup \{s'''\}$ `// If` $e' \in \Sigma_{uo}$
        $M[s', e] \leftarrow \phi(s', e)$
      $\phi(s, \omega'') \leftarrow \phi(s, \omega'') \cup \phi(s', e)$
$Sr \leftarrow \{\{S_0\}\}$; $St \leftarrow \{\{S_0\}\}$
foreach $S = Q_1 \times \ldots \times Q_n$ *in* St **do**
  $St \leftarrow St \setminus \{S\}$
  foreach e *in* Σ_o **do**
    $Q' \leftarrow \emptyset$
    foreach s *in* $Q_1 \cup \ldots \cup Q_n$ **do**
      $Ws \leftarrow \{s\}$
      foreach s' *in* Ws **do**
        $Ws \leftarrow Ws \setminus \{s'\}$
        foreach e' *in* $\{e\} \cup \Sigma_{uo}$ **do**
          if $\delta(s', e') = s''$ **then**
            $\phi(s', e) \leftarrow \phi(s', e) \cup \{s'''\}$ `// If` $e' = e$
            $Ws \leftarrow Ws \cup \{s'''\}$ `// If` $e' \in \Sigma_{uo}$
      foreach $i = 1$ **to** n **do**
        foreach s *in* Q_i **do**
          $Q' \leftarrow Q' \cup \phi(s, e)$
    $S' = Q_1' \times \ldots \times Q_n'$; where $S' \leftarrow \delta_v(S, e)$
    $Sr \leftarrow Sr \cup \{S'\}$; $St \leftarrow St \cup \{S'\}$
$S_v \leftarrow Sr$

Algorithm 2 conducts (ϵ, ξ)-differential privacy verification for a specified number of steps, k. It takes inputs including a verifier V_ω^k, step count k, privacy threshold ϵ, and n initial states. Initially, each state in S_0 has a transition probability of 1. The algorithm iteratively examines state transitions and probability distributions up to step k. A critical aspect of this process is the decay factor ξ, crucial for upholding rigorous privacy constraints. This factor adjusts the influence of the privacy parameter over time, ensuring consistent adherence to privacy norms throughout the algorithm's execution.

Crucially, the algorithm juxtaposes privacy probabilities across all conceivable state pairings. Should any disparity surpass ϵ, the algorithm promptly returns 'false'. This

guarantees a uniform privacy assessment, irrespective of initial state variances, encompassing a predictable future scope. Considering its design and operations, the worst-case computational complexity of Algorithm 2 is gauged at $O(k \times 2^n \times |\Sigma_o| \times n^3 \times |\Sigma_{uo}|)$. This provides an understanding of the resource demands for larger input sizes, crucial for practical implementations.

Algorithm 2: Differential privacy verification over finite steps with ϵ decay

Input: A verifier $V_\omega^k = (S_v, \Sigma_o, \delta_v, S_0)$, a finite step k, an initial positive parameter ϵ, a decay factor ξ, and a list of n initial states $[s_0^1, s_0^2, \ldots, s_0^n]$
Output: True or False

$S_t \leftarrow \{S_0\}$
for $i \leftarrow 1$ **to** n **do**
  $Pr_i(S_0, \epsilon) \leftarrow 1$
for $k' \leftarrow 1$ **to** k **do**
  $\epsilon' \leftarrow \epsilon e^{-\xi k'}$
  foreach $S = Q_1 \times Q_2 \times \ldots \times Q_n \in S_t$ ***and*** $\delta_v(S_0, \omega') = S$ **do**
    Obtain from Algorithm 1 $\phi(s_0^1, \omega\omega'), \phi(s_0^2, \omega\omega'), ..., \phi(s_0^n, \omega\omega')$
    foreach $e \in \Sigma_o$ ***and*** $\delta_v(S, e)$ *is defined* **do**
      foreach $s \in Q_1 \cup Q_2 \cup \ldots \cup Q_n$ **do**
        $\sigma(s, e) \leftarrow \emptyset$
        $\omega_\alpha \leftarrow \epsilon'$
        $W_s \leftarrow \{(\omega_\alpha, s)\}$
        foreach $(\omega_\alpha, s_\alpha) \in W_s$ **do**
          $W_s \leftarrow W_s \setminus \{(\omega_\alpha, s_\alpha)\}$
          foreach $e' \in \{e\} \cup \Sigma_{uo}$ **do**
            $\omega_\alpha \leftarrow \omega_\alpha e'$
            if $\delta(s_\alpha, e') = s_\beta$ ***and*** $e' = e$ **then**
              $\sigma(s, e) \leftarrow \sigma(s, e) \cup \{\omega_\alpha\}$
            if $\delta(s_\alpha, e') = s_\beta$ ***and*** $e' \in \Sigma_{uo}$ **then**
              $W_s \leftarrow W_s \cup \{(\omega_\alpha, s_\beta)\}$
    $P_i \leftarrow 0$ for each $i \in \{1, ..., n\}$
    $s_i \leftarrow s_0^i$ for each $i \in \{1, ..., n\}$
    foreach $s \in S_i$ ***and*** $s \in \sigma(s, e)$ *for each* $i \in \{1, ..., n\}$ **do**
      $P_i \leftarrow P_i + Pr(s|\phi(s_i, \omega\omega')) \times Pr_\sigma(s, u)$
    foreach $i \in \{1, ..., n\}$ **do**
      $Pr_i(S_0, \omega' e) \leftarrow Pr_i(S_0, \omega') \times P_i$
    if *any pair of* $|Pr_i(S_0, \omega' e) - Pr_j(S_0, \omega' e)| \geq \epsilon'$, *for* $i, j \in \{1, ..., n\}$ *and* $i \neq j$ **then**
      return *false*
    $S^* \leftarrow \emptyset$
    foreach $e \in \Sigma_o$ **do**
      $S^* \leftarrow S^* \cup \{\delta_v(S, e)\}$
    $S_t \leftarrow (S_t \setminus \{S\}) \cup S^*$
return *true*

Example 4. *Consider the structure of a probabilistic automaton depicted in Figure 2, where we assume that the initial states are s_1 and s_2, which means $n = 2$. The set of observable events is $\Sigma_o = \{\alpha, \beta, \lambda, \gamma, \mu\}$, while the set of unobservable events is $\Sigma_{uo} = \{\tau\}$. The verifier for the system $\mathcal{G}(s_1)$ and $\mathcal{G}(s_2)$ is illustrated in Figure 3, in which the initial state S_0 is defined as the Cartesian product $S_0 = \{s_1\} \times \{s_2\}$.*

For the state S_1 and the observable event α, we can say that S_1 is the Cartesian product of $\phi(s_1, \alpha)$ and $\phi(s_2, \alpha)$, resulting in $S_1 = \{s_3, s_4\} \times \{s_5\}$. Therefore, $\delta(S_0, \alpha)$ becomes $\{s_3, s_4\} \times \{s_5\}$. Similarly, for the state S_4 and the observable event λ, S_4 is the Cartesian product of $\phi((s_3, s_4), \lambda)$ and $\phi(s_5, \lambda)$, which turns out to be $\{s_4\} \times \emptyset$. Consequently, $\delta(S_1, \lambda)$ is also $\{s_4\} \times \emptyset$. For the state S_5 and the observable event β, S_5 is $\{s_5\} \times \{s_6\}$, and this results from taking the Cartesian product of $\phi(s_3, \beta)$ and $\phi(s_5, \beta)$. Therefore, $\delta(S_1, \beta)$ is $\{s_5\} \times \{s_6\}$. Finally, for the state S_8 and the observable event γ, S_8 is the Cartesian product of $\phi(s_5, \gamma)$ and $\phi(s_6, \gamma)$, yielding $S_8 = \{s_4\} \times \{s_4\}$. This leads us to conclude that $\delta(S_5, \gamma)$ is also $\{s_4\} \times \{s_4\}$.

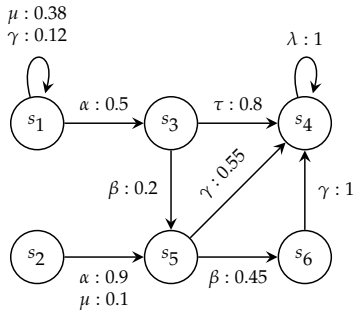

Figure 2. A probabilistic automaton $\mathcal{G}^*$.

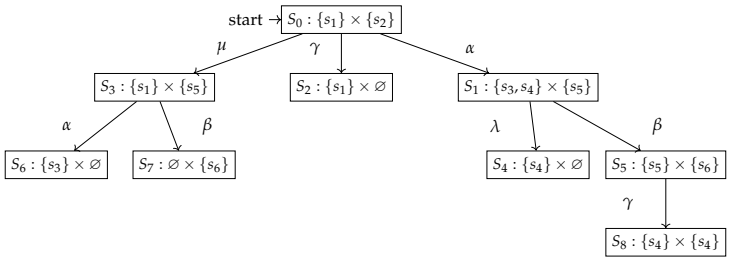

Figure 3. The verifier V_ω^k of probabilistic automaton $\mathcal{G}^*$.

Example 5. *Let us consider the probabilistic automaton structure in Figure 2. Assume two proximate initial states s_1 and s_2 such that $n = 2$. The verifier is shown in Figure 3. We want to verify the ϵ-differential privacy with $\epsilon = 0.13$, decay factor $\xi = 0.05$ and for $k' = 1$. Due to the decay factor, our effective ϵ' for this step is $\epsilon e^{-\xi k'} = 0.13e^{-0.05} \approx 0.124$. For state S_0:*

$$\begin{aligned}
|\Pr_1(S_0,\alpha) - \Pr_2(S_0,\alpha)| &= |\Pr(s_1|\{s_1\}) \times \Pr(\{s_3,s_4\}|\phi(s_1,\alpha)) - \Pr(s_2|\{s_2\}) \times \Pr_\sigma(s_2,\alpha)| \\
&= 0.5/0.5 + (0.5 \times 0.8) + 0.5 \times 0.8/0.5 + (0.5 \times 0.8) - 0.9 \\
&= 0.1 < \epsilon' \approx 0.124; \\
|\Pr_1(S_0,\gamma) - \Pr_2(S_0,\gamma)| &= |\Pr(s_1|\{s_1\}) \times \Pr_\sigma(s_1,\gamma) - \Pr(s_2|\{s_2\}) \times \Pr_\sigma(s_2,\gamma)| \\
&= 0.120 < \epsilon' \approx 0.124; \\
|\Pr_1(S_0,\mu) - \Pr_2(S_0,\mu)| &= |\Pr(s_1|\{s_1\}) \times \Pr_\sigma(s_1,\mu) - \Pr(s_2|\{s_2\}) \times \Pr_\sigma(s_2,\mu)| \\
&= 0.38 - 0.1 = 0.28 > \epsilon' \approx 0.124.
\end{aligned}$$

The system of two automata $\mathcal{G}(s_1)$ and $\mathcal{G}(s_2)$ do not satisfy (ϵ,ξ)-differential privacy at $k' = 1$ with $\epsilon' = 0.124$.

5. Ensuring (ϵ,ξ)-Differential Privacy via Supervisory Control

In the prior section, we introduced the notion of (ϵ,ξ)-differential privacy, where an automaton starting from distinct n initial states produces similar observation likelihoods over a set number of steps. This concept is central to our ongoing discussion. We then delve into supervisory control as a method for ensuring this privacy alignment over a fixed sequence length. Our study further explores the realms of control theory, highlighting a groundbreaking algorithm rooted in probabilistic automata and designed to predict state transitions from observation sequences.

Consider a verifier $V_\omega^k = (S_v, \Sigma_o, \delta_v, S_0)$ and a state $S \in S_v$, where S is an element of the state space S_v and can be expressed as a tuple of multiple state components, i.e., $S = Q_1 \times Q_2 \times \ldots \times Q_n$. We define $S^\#$ as a subset of S_v containing states that can be reached from the initial state S_0 for a specific observation sequence ω'. In essence, $S^\#$ captures the set of states that are intermediate in the transition from S_0 to S through the sequence ω'.

The set $R(S)$ is then introduced to encapsulate all events e that transition the system from state S to some other state S' that is a member of $S^\#$. Specifically, S' represents a

state, analogous to S, but formed from different combinations of state components. Hence, $S' = Q'_1 \times Q'_2 \times \ldots \times Q'_n$ where each Q'_i is a potential state from the original n probabilistic automata. This ensures the system remains within the permissible state transitions defined by $S^{\#}$. Thus, $R(S)$ can be defined as:

$$R(S) = \left\{ e \in \Sigma_o \mid \exists S' \in S^{\#} \setminus \{S\} : \delta_v(S, e) = S' \right\}.$$

This expression underscores the events that, when executed from state S, guide the system to another state within the permissible state transitions defined by $S^{\#}$. Let $\Theta : \Sigma_o \to \mathbb{R}$ be a ranking function defined over the observation events in the context of the verifier V^k_ω. For any event $e \in \Sigma_o$, $\Theta(e)$ designates a unique scalar value, conveying its significance or importance.

This formulation ensures both uniqueness, where for any two distinct events $e_1, e_2 \in \Sigma_o$, if $e_1 \neq e_2$ then $\Theta(e_1) \neq \Theta(e_2)$, and a total order such that for any two events e_1 and e_2 in Σ_o, the relationships $\Theta(e_1) < \Theta(e_2)$, $\Theta(e_1) > \Theta(e_2)$, or $\Theta(e_1) = \Theta(e_2)$ are established.

Once the events are ranked by their $\Theta(e)$ values, each event e is assigned an index value $T(S, e)$, where $T : S_v \times \Sigma_o \to \mathbb{N}^+$ is a mapping function. In this context, for states $S = Q_1 \times Q_2 \times \cdots \times Q_n \in S_v$ and $S' = Q'_1 \times Q'_2 \times \cdots \times Q'_n \in S^*$, the set of all events that transition the system from S' to S is denoted as $E(S', S) = \{e \in \Sigma_o \mid \delta_v(S', e) = S\}$.

Consequently, the events in $R(S)$ are systematically sorted based on their respective $\Theta(e)$ values in ascending order, ensuring a structured and meaningful sequence for subsequent operations, reflecting both the probabilistic properties and temporal nuances of the events within V^k_ω.

For each event e belonging to $E(S', S)$, we define $A(S', S, e)$ as a column vector with $|R(S')|$ dimensions, exclusively consisting of zeros and ones. Additionally, we define a binary scalar $A(S', S, e)[v]$ in the following manner:

$$A(S', S, e)[c] = \begin{cases} 1, & \text{if } v = T(S', e) \\ 0, & \text{if } v \neq T(S', e). \end{cases}$$

With v being a positive integer such that $1 \leq v \leq |R(S')|$, the matrix $A(S', S)$ can be viewed as an expanded form of $A(S', S, e)$ for every event e in $E(S', S)$, arranged in ascending order according to the value of $T(S', e)$. Given a probabilistic automaton structure $\mathcal{G} = (S, \Sigma, \delta, \rho)$ with proximate initial states $s_0^{(1)}, s_0^{(2)}, \ldots, s_0^{(n)}$, and an observation sequence $\omega \in \Sigma_o^*$, we define $V^k_\omega = (S_v, \Sigma_o, \delta_v, S_0)$ as the verification mechanism. For a compound state $S = Q_1 \times Q_2 \times \ldots \times Q_n$ in S_v, and given $\delta_v(S_0, \omega') = S$, any of the row vectors $C_{\omega'}(S_1|S), C_{\omega'}(S_2|S), \ldots, C_{\omega'}(S_n|S)$ is with the dimensioning of $|R(S)|$. For each event $e \in R(S)$ and index $T(S, e) \in \{1, 2, \ldots, |R(S)|\}$, the following relationship holds for $i \in \{1, 2, \ldots, n\}$:

$$C_{\omega'}(S_i|S)[T(S, e)] = \sum_{s \in S_i} \sum_{s' \in \sigma(s,e)} \left[\Pr_\sigma(s|\phi(s_0^{(i)}, \omega\omega')) \times \Pr_\sigma(s', u) \right].$$

Given a probabilistic automaton $\mathcal{G} = (S, \Sigma, \delta, \rho)$ with n initial states $s_0^{(1)}, s_0^{(2)}, \ldots, s_0^{(n)}$ and an observation $\omega \in \Sigma_o^*$, let $V^k_\omega = (S_v, \Sigma_o, \delta_v, S_0)$ serve as the verificationer. For a compound state $S = Q_1 \times Q_2 \times \cdots \times Q_n$ residing in the state space S_v, and given $\delta_v(S_0, \omega') = S$, we define $A^{\omega'}(S_i|S)$ as the corresponding probabilistic matrix for the substate S_i, the composite state S, and the observation sequence $\omega' \in \Sigma_o^*$, where i is an index drawn from the set $\{1, 2, \ldots, n\}$. The definition of $A^{\omega'}(S_i|S)$ unfolds in two scenarios:

First, in the event that $S^* = \emptyset$, the matrix $A^\varepsilon(S_i|S)$ is identified directly as $C_\varepsilon(S_i|S)$;

Second, when $S^{\#} \neq \emptyset$, for any $S' = Q'_1 \times Q'_2 \times \cdots \times Q'_n$ belonging to $S^{\#}$ such that $\delta_v(S_0, \omega'') = S'$, $\delta_v(S', e) = S$, and $\omega'' e = \omega'$, the following relation holds:

$$A^{\omega'}_m(S_i|S) = \left[A^{\omega''}(S'_i|S') \times A(S', S) \right]^T [:][m] \times C_{\omega'}(S_i|S);$$

Subsequently, the exhaustive probabilistic matrix $A^{\omega'}(S_i|S)$ is synthesized as

$$A^{\omega'}(S_i|S) = \left[(A_1^{\omega'}(S_i|S))^T|\cdots|(A_h^{\omega'}(S_i|S))^T\right]^T;$$

where $h = \mathbb{N}^r(A^{\omega''}(S_i'|S'))$; this relationship reflects the intricate structure and hierarchy inherent in the probabilistic matrices, emphasizing a crucial characteristic, possibly its dimensionality or structural depth.

Within this setup, h is deduced by elevating $\mathbb{N}$ to the power of r. It is conjectured that r represents the rank or a distinct inherent trait of the matrix $A^{\omega''}(S_i'|S')$. This formulation provides a foundation for the ensuing algorithmic stages, imparting a computational perspective to the entire procedure. A clear linkage between m and h exists, with m being an element of the set $\{1, 2, \ldots, h\}$. The strategy for computing these probabilistic matrices for each state in the verifier is encapsulated in Algorithm 3, demonstrating a computational complexity of $O\Big(|S_v|^2 \times (|\Sigma_o| + |R(S')| + |R(S)| + n \times |S_i|)\Big)$.

Algorithm 3: Probability matrices determination

Input: A verifier $V_\omega^k = (S_v, \Sigma_o, \delta_v, S_0)$ and a state $S = Q_1 \times Q_2 \times ... \times Q_n$ with $\delta_v(S_0, \omega') = S$
Output: Probability matrices $A^{\omega'}(S_1|S)$ and $A^{\omega'}(S_2|S)$ ... $A^{\omega'}(S_m|S)$

$S^\# \leftarrow \emptyset$ **foreach** $S_a \in S_v$ **do**
  if $\delta_v(S_0, \omega'') = S_a$ ***and*** $\delta_v(S_a, e) = S$ ***and*** $\omega'' e = \omega'$ **then**
    $S^\# \leftarrow S^\# \cup \{S_a\}$;
foreach $S' = Q_1' \times Q_2' \times ... \times Q_n' \in S^\#$ **do**
  $A(S', S) \leftarrow []$; $R(S') \leftarrow \emptyset$; $R(S) \leftarrow \emptyset$ **foreach** $e \in \Sigma_o$ **do**
    foreach $S_a \in \{S', S\}$ **do**
      if $\delta_v(S_a, e)$ *is defined* ***and*** $\delta_v(S_a, e) \neq S_a$ **then**
        $R(S_a) \leftarrow R(S_a) \cup \{e\}$
  foreach $e \in \Sigma_o$ **do**
    if $\delta_v(S', e) = S$ **then**
      for $a = 1$ **to** $|R(S')|$ **do**
        if $a = T(S', e)$ **then**
          $A(S', S, e)[a][1] \leftarrow 1$
        else
          $A(S', S, e)[a][1] \leftarrow 0$
      $A(S', S) \leftarrow [A(S', S)|A(S', S, e)]$
      foreach $i \in \{1, ..., n\}$ **do**
        $B_i \leftarrow A^{\omega''}(S_i'|S') \times A(S', S)$
        Obtain from Algorithm 1 $\phi(s_0^1, \omega\omega'), \phi(s_0^2, \omega\omega'), ..., \phi(s_0^n, \omega\omega')$
        foreach $e \in R(S)$ **do**
          foreach $s \in S_i$ **do**
            compute $\sigma(s, e)$ by Algorithm 2
            $C_{\omega'}(S_i|S)[1][T(S, e)] \leftarrow \sum_{s \in \sigma(s,e)} \sum_{s \in S_i} [Pr(s|\phi(s_0^i, \omega\omega')) \times Pr_\sigma(s, u)]$
        $t \leftarrow \mathbb{N}^r\left(A^{\omega''}(S_i'|S')\right)$ **for** $m = 1$ **to** t **do**
          $A_m^{\omega'}(S_i|S) \leftarrow B_i^T[:, m] \times C_{\omega'}(S_i|S)$
        $A^{\omega'}(S_i|S) \leftarrow [A_1^{\omega'}(S_i|S)^T|...|A_t^{\omega'}(S_i|S)^T]^T$

Example 6. *Consider the probabilistic automaton structure represented in Figure 2 and its corresponding verifier illustrated in Figure 3, in which the given parameters are* $\Sigma_o = \{\alpha, \beta, \lambda, \gamma, \mu\}$ *and* $\Sigma_{uo} = \{\tau\}$. *We define the function* Θ *as:* $\Theta(\alpha) = 1$, $\Theta(\beta) = 2$, $\Theta(\lambda) = 3$, $\Theta(\gamma) = 4$, *and* $\Theta(\mu) = 5$. *For* S_0, *we obtain* $E(S_0, S_1) = \{\alpha\}$, $A(S_0, S_1) = [A_\alpha]$ *with* $A_\alpha = (1, 0, 0)^T$ *and* $E(S_0, S_3) = \{\mu\}$, $A(S_0, S_3) = [A_\mu]$ *with* $A_\mu = (0, 0, 1)^T$. *For* S_2, *we obtain* $R(S_5) = \{\gamma\}$ *and* $T(S_5, \gamma) = 1$. *Thus,* $C_{\omega'}(\{s_5\}|S_5) = Pr_\sigma(s_5, \gamma) = 0.55$ *and* $C_{\omega'}(\{s_6\}|S_5) = Pr_\sigma(s_6, \gamma) = 1$, *where* $\omega' = \{\alpha\beta\}$.

Example 7. *Consider the verification depicted in Figure 3. For the initial state S_0, we have $A^{\varepsilon}(\{s_1\}|S_0) = (0.5, 0.12, 0.38)$ and $A^{\varepsilon}(\{s_2\}|S_0) = (0.9, 0, 0.1)$. For state S_3: $A^{\varepsilon}(\{s_1\}|S_0) \times A(S_0, S_3) = 0.38$; $A^{\varepsilon}(\{s_2\}|S_0) \times A(S_0, S_3) = 0.1$. For an observable event $\omega' = \mu$, $A^{\omega'}(\{s_1\}|S_3) = (0.38)^T[:][2] \times C_{\omega'}(\{s_1\}|S_3) = (0.19, 0)^T$ and $A^{\omega'}(\{s_5\}|S_3) = (0.1)^T[:][2] \times C_{\omega'}(\{s_5\}|S_3) = (0, 0.045)^T$.*

Within the intricate domain of probabilistic automata, the introduction of *fake events* represents a paradigm shift in how systems interact with their observable events. This is not a superficial addition; rather, it is a calculated stratagem where these events, ingeniously derived from the set Σ_o of genuine observable events, play a pivotal role in ensuring system integrity and privacy [37]. The origin of these fake events can be traced back to the enforcement mechanism, which incorporates an insertion function designed to embed any event within Σ_o seamlessly. While on the surface, these inserted events might mirror the observable ones, their underlying design is distinct. The crucial aspect lies in their deliberate ambiguity; they mirror the observable events with such precision that they become indistinguishable upon a cursory examination. This resemblance is not coincidental but serves a pivotal role in the broader system dynamics.

The aforementioned duality serves a higher purpose, particularly in the domain of differential privacy. The insertion of fake events is not merely a technique for enhancing system dynamics; it acts as a vital control strategy. When the system fails to achieve (ε, ξ)-differential privacy through conventional means, these fake events step in to refine the transition probabilities of the genuine events within Σ_o, ensuring the attainment of the desired (ε, ξ)-differential privacy threshold. Such innovation highlights the sophistication and flexibility of probabilistic automata theory, where privacy safeguards and operational effectiveness are adeptly intertwined.

The supervisory control system in question operates based on observed events Σ_o. In addition to these genuine events, we introduce fabricated events, denoted by Σ_{fake}. Each fabricated event is a simulation derived from its Σ_o counterpart. Collectively, the universe of all events is given by $\Sigma = \Sigma_o \bigcup \Sigma_{\text{fake}} \bigcup \Sigma_{\text{uo}}$. The matrix D is a configuration of size $n \times n$, with n indicating the system's state count. An element $D[i][j]$ delineates the difference in actions executed at states Q_i and Q_j in response to a specific observation ω'. This relationship is formally described by $D[i][j] = A^{\omega'}(Q_i \mid S) - A^{\omega'}(Q_j \mid S)$. To incorporate (ϵ, ξ)-differential privacy, we leverage an insertion function, E, which maps a state and a genuine event to a corresponding fabricated event. Formally, $\Phi : S \times \Sigma_o \to S \times \Sigma_{\text{fake}}$. The execution of this function results in $\Phi(s, e) = (s', E(s, e))$, with s' determined by $s' = \delta(s, e)$. This function's application adjusts the event probability distribution. For instance, triggering an artificial event, e_{fake}, at state s modulates the event probabilities as $\rho'(s, e) = \rho(s, e) - \rho(s, e_{\text{fake}}) \times \rho(s, e)$.

In the context of a supervisory control system within a probabilistic automaton, the process of refining transition probabilities through the insertion of fake events plays a pivotal role in adhering to differential privacy constraints. Consider a system where the transition probability from an initial state by a particular observation sequence ω' is formulated as $A^{\omega'}(Q \mid S) = Pr_{\sigma}(s_0, \omega'' e) \times \rho(\delta(s_0, \omega''), e)$, with $\omega' = \omega'' e$, where $\omega'' \in \Sigma_o^*$ and $e \in \Sigma_o$. This formulation encapsulates the probability of the sequence $\omega'' e$ occurring from the initial state s_0 and the probability of transitioning to the next state due to event e.

The introduction of a fake event, executed by the supervisory function E, fundamentally alters this transition probability. Post-insertion, the probability of the transition by the observation sequence ω' is refined to $A^{\omega'}(Q \mid S) = Pr_{\sigma}(s_0, \omega'' e) \times \rho(\delta(s_0, \omega''), e) \times (1 - \rho(\delta(s_0, \omega''), e_{\text{fake}}))$. The inclusion of the term $(1 - \rho(\delta(s_0, \omega''), e_{\text{fake}}))$, which lies between 0 and 1, ensures a reduction in $A^{\omega'}(Q \mid S)$ subsequent to the insertion of the fake event.

This reduction plays a crucial role in aligning the system with differential privacy constraints. When the inequality $|(A^{\omega'}(Q_i \mid S) - (A^{\omega'}(Q_j \mid S)| \geq \epsilon'$ holds, indicating a potential breach of privacy thresholds, the insertion of the fake event with a predetermined probability effectively diminishes the left-hand side of the inequality by a fac-

tor of $(1-\rho(\delta(s_0,\omega''),e_{\text{fake}}))$. This decrement contributes significantly to reducing the differences in probabilities of sequences resulting from the initial states (s_0^i, s_0^j), thus facilitating the achievement of the differential privacy constraint. Therefore, the strategic insertion of fake events emerges as a vital mechanism for refining transition probabilities in a manner that upholds the principles of differential privacy within the framework of probabilistic automata.

Theorem 2. *Given a supervisory control system formulated by the verifier $V_k^\omega = (S_v, \Sigma_o, \delta_v, S_0)$. The introduction of a fake event mechanism, governed by the insertion function E, that operates within the (ϵ, ξ)-differential privacy boundary will ensure convergence towards a refined system state such that the differential between state transitions, quantified by matrix D, remains beneath the threshold ϵ'.*

Proof. Starting with the differential matrix representation for each pair of states (Q_i, Q_j) in our system, the differential is defined as $D[i][j] = A^{\omega'}(Q_i \mid S) - A^{\omega'}(Q_j \mid S)$, capturing the initial difference between the states in light of a particular observation ω. For any given state s and an event e from Σ_o, the insertion function is described as $E : S \times \Sigma_o \to \Sigma_{\text{fake}}$, producing a fake event $e_{\text{fake}} = E(s,e)$. With this new event, the state transition dynamics are altered: the state transition probabilities evolve to $\rho'(s,e) = \rho(s,e) - \rho(s,e_{\text{fake}}) \times \rho(s,e)$, ensuring the recalculated differential remains constrained within the threshold $|D'[i][j]| \leq \epsilon'$. Assuming that the introduction of the fake event modifies the state transition mechanisms, the system undergoes iterative invocations of the insertion function to guarantee all differentials in matrix D adhere to $\forall i, j : |D[i][j]| \leq \epsilon'$.

Given a maximum bound of iterations, denoted as k, combined with the consistent adjustment via the insertion function E, it is implied that convergence occurs within these iterations. Crucially, each introduction of a fake event acts as a remedial measure, aligning the system's dynamics closer to the (ϵ, ξ)-differential privacy conditions. Through our established iterations, and by ensuring differentials remain within the specified boundary, the system not only guarantees state transition convergence but also maintains the constraints of differential privacy. Thus, the theorem is affirmed, elucidating how the supervisory control system, using the intricate balance of fake event insertion, converges toward refined states while upholding (ϵ, ξ)-differential privacy. □

Transitioning to the algorithmic realm, Algorithm 4 stands at the intersection of computer science, data analytics, and (ϵ, ξ)-differential privacy.

Algorithm 4 is meticulously crafted to ensure that whenever variances within matrix D surpass a predefined threshold ϵ', a fake event, e_{fake}, is integrated into Σ_o.

This intricate mechanism ensures a balance between maintaining (ϵ, ξ)-differential privacy mandates and upholding the system's dynamic transitions. The core of this algorithm focuses on extracting the value a_i from the observation matrix $A'(Q_i \mid S)$ while concurrently formulating z_i using $z_i \leftarrow \sum_{s \in Q_i}[Pr(s|\phi(s_0^i, \omega')) \times \rho(s,e)]$. Remarkably, despite its sophistication, Algorithm 4 boasts a computational efficiency with complexity of $O(k \times |S_v| \times n^3 \times |\Sigma_o|)$, thereby encapsulating the essence of modern supervisory control-bridging privacy safeguards with system dynamism.

Theorem 3. *For the algorithm "Iterative supervisory control refinement for multiple initial states", given a verifier $V_\omega^k = (S_v, \Sigma_o, \delta_v, S_0)$ and a positive integer k, the algorithm converges to a refined set of states in at most k iterations, ensuring that the differential between state transitions, as indicated by matrix D, remains below the threshold ϵ'.*

Proof. Initiating with St set to the initial state S_0, the algorithm iteratively refines states, bounded by a maximum of k iterations. The matrix D captures state differentials, and the algorithm ensures that none of its entries surpass ϵ'. If an entry in D breaches this threshold, a "fake event" is introduced, adjusting state transitions to bring the differential back within bounds.

The algorithm's iterations (lines 10–18) either keep differentials in D under ϵ' or apply corrective mechanisms to achieve this. Hence, within k iterations, the algorithm guarantees that state transition differentials adhere to the threshold ϵ'. □

Algorithm 4: Iterative supervisory control refinement for multiple initial states

Data: A verifier $V_\omega^k = (S_v, \Sigma_o, \delta_v, S_0)$ and a positive integer k
Result: Refined set of states
1 Initialize $S_t \leftarrow \{S_0\}$; $S_m \leftarrow \emptyset$
2 **for** $k' = 1$ **to** k **do**
3 Update ϵ'
4 **foreach** $S = Q_1 \times Q_2 \times ... \times Q_n$ *in* S_t *and* $\delta_v(S_0, \omega') = S$ **do**
5 Initialize D of size $[n][n]$
6 $S^* \leftarrow \emptyset$
7 **for** $i = 1$ **to** n **do**
8 **for** $j = i + 1$ **to** n **do**
9 $D[i][j] \leftarrow A^{\omega'}(Q_i|S) - A^{\omega'}(Q_j|S)$
10 **foreach** $D[i][j][d]$ **do**
11 **if** $|D[i][j][d]| > \epsilon'$ **then**
12 **foreach** s *in* $Q_i \cup Q_j$ *such that* $T(S, e) = j$ **do**
13 $(s', e_{\text{fake}}) \leftarrow \Phi(s, E(s, e))$
14 $Q' \leftarrow Q' \cup \{s'\}$
15 $S' = Q_i' \times Q_j'$; which yields $S' \leftarrow \delta(S, e)$
16 Update $A^{\omega'}(Q_i|S)$ and $A^{\omega'}(Q_j|S)$
17 go to line 10
18 Obtain from Algorithm 1 $\phi(s_0^1, \omega\omega'), \phi(s_0^2, \omega\omega'), ..., \phi(s_0^n, \omega\omega')$
19 **foreach** e *in* Σ_o *and* $\delta_v(S, e) = S$ **do**
20 **for** $i = 1$ **to** n **do**
21 Calculate weight z_i based on $\phi(s_i^0, \omega\omega')$ and $\rho(s, e)$
22 **for** $n = 1$ **to** $k - k'$ **do**
23 **foreach** $a_i = A^{\omega'}(Q_i|S)[a][b]$ **do**
24 **for** $j = 1$ **to** n **do**
25 **if** $|z_n \times a_i - z_j \times a_j| > \epsilon'$ **then**
26 **foreach** s *in* $Q_1 \cup Q_2 \cup ... \cup Q_n$ **do**
27 Call subroutine for adding fake event
28 **go to** line 19
29 **foreach** e *in* Σ_o *and* $\delta_v(S, e) \neq S$ **do**
30 $S^* \leftarrow S^* \cup \{\delta_v(S, e)\}$
31 $S_t \leftarrow S_t \setminus \{S\}$, $S_m \leftarrow S_m \cup (S^* \setminus \{S\})$, $S_t \leftarrow S_m$, $S_m \leftarrow \emptyset$

6. Numerical Case Study

This section presents a numerical case study that precisely illustrates the proposed methodology. The case study confirms the methodology's effectiveness and applicability to discrete event systems modeled by probabilistic automata. A DES represented by probabilistic automata is shown in Figure 4, where $\Sigma_o = \{\alpha, \beta, \lambda, \gamma, \mu\}$, $\Sigma_{uo} = \{\tau\}$, $\Sigma_{fake} = \{\alpha', \beta', \lambda', \gamma', \mu'\}$, and the initial states are: $\{s_0, s_1, s_2\}$. The verifier of this system is represented in Figure 5. Let $\Theta(\alpha) = 1$, $\Theta(\beta) = 2$, $\Theta(\lambda) = 3$, $\Theta(\gamma) = 4$, $\Theta(\mu) = 5$, $\epsilon = 0.14$, and $\xi = 0.01$.

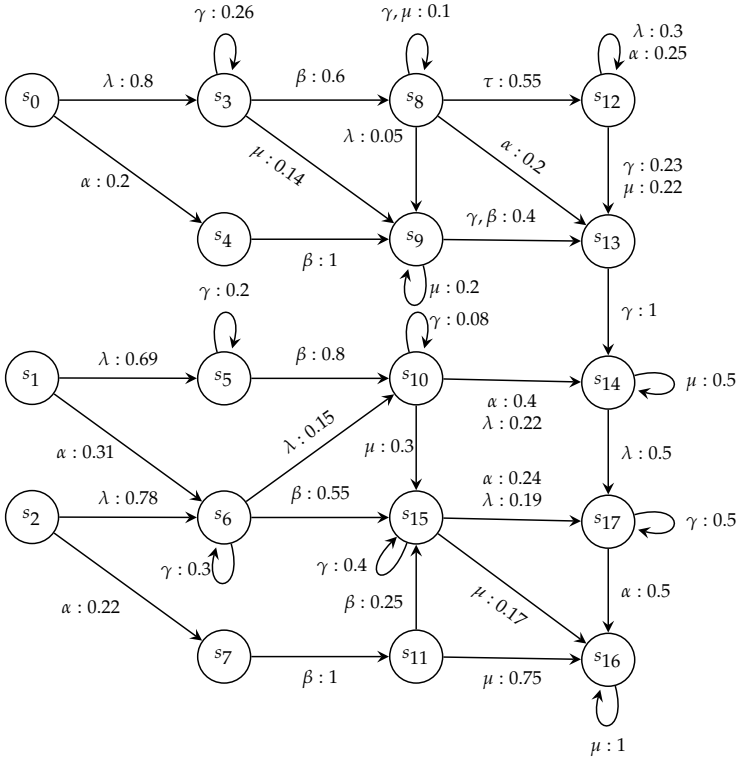

Figure 4. A probabilistic automaton $\mathcal{G}^{\#}$.

For $k' = 1$ and state S_0 in $V^k_{\omega'}$: $\epsilon'_1 = \epsilon e^{-0.01} = 0.139$.

$$T(S_0, \alpha) = 1, T(S_0, \lambda) = 2.$$
$$A^{\varepsilon}(\{s_0\}|S_0) = (0.2, 0.8);$$
$$A^{\varepsilon}(\{s_1\}|S_0) = (0.31, 0.69);$$
$$A^{\varepsilon}(\{s_2\}|S_0) = (0.22, 0.78).$$

Probability differences are:

$$(s_0, s_1) = (0.110, 0.110);$$
$$(s_1, s_2) = (0.090, 0.090);$$
$$(s_2, s_0) = (0.020, 0.020).$$

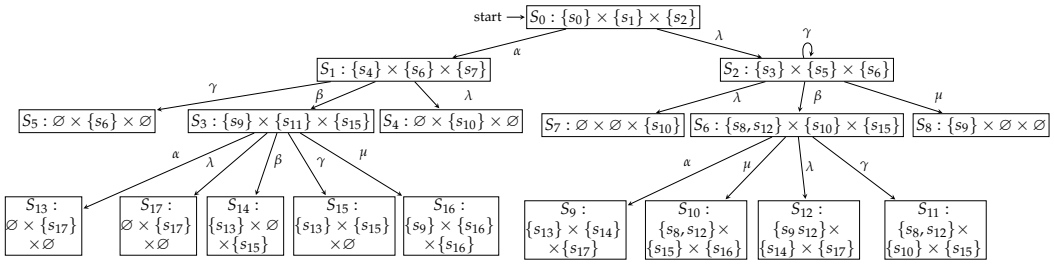

Figure 5. The verifier $V^k_{\omega'}$ for $k' = 3$ of probabilistic automaton $\mathcal{G}^{\#}$.

For $k' = 2 : \epsilon'_2 = \epsilon e^{-0.02} = 0.136$. For state S_1:

$$T(S_1, \beta) = 1, T(S_1, \lambda) = 2, T(S_1, \gamma) = 3.$$
$$A^{\varepsilon}(\{s_0\}|S_0) \times A(S_0, S_1) = 0.2;$$
$$A^{\varepsilon}(\{s_1\}|S_0) \times A(S_0, S_1) = 0.31;$$
$$A^{\varepsilon}(\{s_2\}|S_0) \times A(S_0, S_1) = 0.22.$$

Then:

$$A^{\varepsilon}(\{s_4\}|S_1) = (0.2, 0, 0);$$
$$A^{\varepsilon}(\{s_6\}|S_1) = (0.171, 0.047, 0.093);$$
$$A^{\varepsilon}(\{s_7\}|S_1) = (0.220, 0, 0).$$

Probability differences are:

$$(s_4, s_6) = (0.029, 0.047, 0.093);$$
$$(s_6, s_7) = (0.049, 0.047, 0.093);$$
$$(s_7, s_4) = (0.020, 0, 0).$$

For $k' = 2$ and state S_2:

$$T(S_2, \beta) = 1, T(S_2, \lambda) = 2, T(S_2, \mu) = 3.$$
$$A^{\varepsilon}(\{s_0\}|S_0) \times A(S_0, S_2) = (0.2, 0.8) \times (0, 1)^T = 0.8;$$
$$A^{\varepsilon}(\{s_1\}|S_0) \times A(S_0, S_2) = (0.31, 0.69) \times (0, 1)^T = 0.69;$$
$$A^{\varepsilon}(\{s_2\}|S_0) \times A(S_0, S_2) = (0.22, 0.78) \times (0, 1)^T = 0.78.$$

For s_3:

$$A^{\lambda}(\{s_3\}|S_2) = 0.8 \times C_{\lambda}(\{s_3\}|S_2) = 0.8(0.6, 0, 0.14) = (0.480, 0, 0.112);$$
$$\rho(s_3, \gamma) = 0.26;$$
$$(\rho(s_3, \gamma))^1 \times A^{\lambda}(\{s_3\}|S_2) = 0.26 \times (0.48, 0, 0.11) = (0.125, 0, 0.029);$$
$$(\rho(s_3, \gamma))^2 \times A^{\lambda}(\{s_3\}|S_2) = (0.26)^2 \times (0.48, 0, 0.11) = (0.032, 0, 0.007).$$

For s_5:

$$A^{\lambda}(\{s_5\}|S_2) = 0.69 \times C_{\lambda}(\{s_5\}|S_2) = 0.69(0.8, 0, 0) = (0.552, 0, 0);$$
$$\rho(s_5, \gamma) = 0.2;$$
$$(\rho(s_5, \gamma))^1 \times A^{\lambda}(\{s_5\}|S_2) = 0.2 \times (0.55, 0, 0) = (0.110, 0, 0);$$
$$(\rho(s_5, \gamma))^2 \times A^{\lambda}(\{s_5\}|S_2) = (0.2)^2 \times (0.55, 0, 0) = (0.004, 0, 0).$$

For s_6:

$$A^{\lambda}(\{s_6\}|S_2) = 0.8 \times C_{\lambda}(\{s_6\}|S_2) = 0.78(0.55, 0.15, 0) = (0.429, 0.117, 0);$$
$$\rho(s_6, \gamma) = 0.3;$$
$$(\rho(s_6, \gamma))^1 \times A^{\lambda}(\{s_6\}|S_2) = 0.3 \times (0.43, 0.12, 0) = (0.129, 0.036, 0);$$
$$(\rho(s_6, \gamma))^2 \times A^{\lambda}(\{s_6\}|S_2) = (0.3)^2 \times (0.43, 0.12, 0) = (0.039, 0.011, 0).$$

For $k' = 3$: $\epsilon'_3 = \epsilon e^{-0.03} = 0.132$. For state S_3:

$T(S_3, \alpha) = 1, T(S_3, \beta) = 2, T(S_3, \lambda) = 3, T(S_3, \gamma) = 4, T(S_3, \mu) = 5.$
$A^{\alpha}(\{s_4\}|S_1) \times A(S_1, S_3) = (0.2, 0, 0) \times (1, 0, 0)^T = 0.2;$
$A^{\alpha}(\{s_6\}|S_1) \times A(S_1, S_3) = (0.171, 0.047, 0.093) \times (1, 0, 0)^T = 0.171;$
$A^{\alpha}(\{s_7\}|S_1) \times A(S_1, S_3) = (0.22, 0, 0) \times (1, 0, 0)^T = 0.22.$
$A^{\alpha\beta}(\{s_9\}|S_3) = 0.2 \times C_{\alpha\beta}(\{s_9\}|S_3) = 0.2 \times (0, 0.4, 0, 0.4, 0.2) = (0, 0.080, 0, 0.080, 0.040);$
$A^{\alpha\beta}(\{s_{15}\}|S_3) = 0.17 \times C_{\alpha\beta}(\{s_{15}\}|S_3) = 0.17 \times (0.24, 0, 0.19, 0.4, 0.17);$
$= (0.041, 0, 0.032, 0.068, 0.029);$
$A^{\alpha\beta}(\{s_{11}\}|S_3) = 0.22 \times C_{\alpha\beta}(\{s_{11}\}|S_3) = 0.22 \times (0, 0.25, 0, 0, 0.75);$
$= (0, 0.055, 0, 0, 0.165).$

Probability differences are:

$$(s_9, s_{15}) = (0.041, 0.080, 0.032, 0.012, 0.011);$$
$$(s_{15}, s_{11}) = (0.041, 0.055, 0.032, 0.068, 0.136);$$
$$(s_{11}, s_9) = (0, 0.025, 0, 0.080, 0.125).$$

For $k' = 3$ and state S_6:

$$T(S_6, \alpha) = 1, T(S_6, \lambda) = 2, T(S_6, \gamma) = 3, T(S_6, \mu) = 4.$$

We then compute:

$$A^{\lambda}(\{s_3\}|S_2) \times A(S_2, S_6) = 0.480;$$
$$A^{\lambda}(\{s_5\}|S_2) \times A(S_2, S_6) = 0.552;$$
$$A^{\lambda}(\{s_6\}|S_2) \times A(S_2, S_6) = 0.429.$$

Here, the probability of choosing either (s_8 or s_{12}) by the $\omega' = \lambda\beta$ is conditional such that:

$$Pr(\ s_{12}|\phi(s_0, \lambda\beta)) = \frac{0.48 \times 0.55}{0.48 \times 0.55 + 0.48} = 0.355;$$
$$\Pr(\ s_8|\phi(s_0, \lambda\beta)) = \frac{0.48}{0.48 \times 0.55 + 0.48} = 0.645.$$

Next, we have:

$$A^{\lambda\beta|}(\{s_8, s_{12}\}|S_6) = 0.48 \times C_{\lambda\beta}(\{s_8, s_{12}\}|S_6) =$$
$$A^{\lambda\beta|}(\{s_8, s_{12}\}|S_6) = (0.218, 0.139, 0.146, 0.143);$$
$$A^{\lambda\beta}(\{s_{10}\}|S_6) = (0.221, 0.121, 0.044, 0.165);$$
$$A^{\lambda\beta}(\{s_{15}\}|S_6) = (0.103, 0.081, 0.172, 0.073).$$

Probability differences are:

$$(\{s_8, s_{12}\}, s_{10}) = (0.003, 0.018, 0.102, 0.022);$$
$$(s_{10}, s_{15}) = (0.118, 0.040, 0.128, 0.920);$$
$$(s_{15}, \{s_8, s_{12}\}) = (0.115, 0.058, 0.026, 0.070).$$

Based on the numerical investigation conducted, the automaton $\mathcal{G}^{\#}$ satisfies the (ϵ, ξ)-differential privacy condition for $k' = 1$ with $\epsilon'_1 = 0.139$ and for $k' = 2$ with $\epsilon'_2 = 0.136$. However, the (ϵ, ξ)-differential privacy condition is not met for $k' = 3$ with $\epsilon'_3 = 0.132$ between the pair (s_{15}, s_{11}). For the state s_{11}, triggering the fake μ' via supervisory control such that: $\Phi(s_{11}, \mu') = (s_{11}, \mu')$, will lead to redistribution of events probabilities as follows: $(0, 0.225, 0, 0, 0.675, 0.1)$. By triggering μ' at state s_{15} such that $\Phi(s_{15}, \mu') = (s_{15}, \mu')$, the adjusted transitions probabilities are as follows: $(0.216, 0, 0.171, 0.36, 0.153, 0.1)$. Now, we

are computing the transformed probabilities, taking into account the effect of the triggered fake event μ':

$$A^{\alpha\beta}(\{s_{11}\}|S_3) = 0.22 \times (0, 0.225, 0, 0, 0.675) = (0, 0.050, 0, 0, 0.149, 0.022);$$
$$A^{\alpha\beta}(\{s_{15}\}|S_3) = 0.171 \times (0.216, 0, 0.171, 0.36, 0.153)$$
$$= 0.037, 0, 0.029, 0.062, 0.026, 0.017);$$
$$A^{\alpha\beta}(\{s_9\}|S_3) = 0.2 \times (0, 0.4, 0, 0.4, 0.2, 0) = (0, 0.080, 0, 0.080, 0.040).$$

Based on the above, the supervisory mechanism successfully enforces the (ϵ, ξ)-differential privacy for $k' = 3$ with $\epsilon'_3 = 0.132$.

7. Conclusions

Our study marks a significant advancement in probabilistic automata, introducing a verification protocol aimed at protecting initial states. Utilizing advanced mathematical methods, this protocol evaluates privacy risks in event sequences and incorporates a supervisory control to maintain privacy without sacrificing system functionality. Looking ahead, we plan to explore the integration of probabilistic automata with dynamic concealment frameworks, focusing on adaptability and responsiveness in various system environments.

Despite these advancements, we recognize areas needing further exploration and improvement. Managing the complexity of multiple initial states in probabilistic automata is challenging, particularly regarding scalability and efficiency in larger systems. The reliance on precise observation sequences is another critical aspect, as any inaccuracies could undermine the reliability of our privacy assurances. The resource-intensive nature of our approach also necessitates consideration, especially in settings with limited resources. Additionally, enhancing the model's adaptability to dynamic systems with frequently changing initial states and behaviors is a crucial future direction. Finally, broadening the methodology's applicability to diverse systems and domains remains an essential goal.

Thus, while our research establishes a solid foundation for differential privacy in discrete event systems using probabilistic automata, it also underscores the need for continuous advancements in complexity management, observation accuracy, resource optimization, adaptability, and application scope. These areas will be central to our ongoing research efforts in this evolving field.

Author Contributions: Conceptualization, Z.L.; methodology, T.A.A.-S.; software, G.Z.; validation, M.A.E.-M.; formal analysis, M.S.; investigation, T.A.A.-S. and G.Z.; resources, M.A.E.-M. and M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the Key Technology R&D Program of Henan Province of China (Grant No. 232102220060), National Natural Science Foundation of China (Grant No. 62103349), and the Special Fund for Scientific and Technological Innovation Strategy of Guangdong Province (Grant No. 2022A0505030025). The authors present their appreciation to King Saud University for funding this research through Researchers Supporting Program number (RSPD2023R704), King Saud University, Riyadh, Saudi Arabia.

Data Availability Statement: The experimental data used in this paper can be obtained by contacting the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DESs Discrete Event Systems

References

1. Al-Makhlafi, M.; Gu, H.; Almuaalemi, A.; Almekhlafi, E.; Adam, M.M. RibsNet: A scalable, high-performance, and cost-effective two-layer-based cloud data center network architecture. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 1676–1690. [CrossRef]
2. Rao, P.S.; Satyanarayana, S. Privacy-preserving data publishing based on sensitivity in context of Big Data using Hive. *J. Big Data* **2018**, *5*, 20. [CrossRef]
3. Jain, P.; Gyanchandani, M.; Khare, N. Big data privacy: A technological perspective and review. *J. Big Data* **2016**, *3*, 472–496. [CrossRef]
4. Yao, L.; Chen, Z.; Hu, H.; Wu, G.; Wu, B. Sensitive attribute privacy preservation of trajectory data publishing based on l-diversity. *Distrib. Parallel Databases* **2020**, *39*, 785–811. [CrossRef] [PubMed]
5. Zhang, B.; Lin, J.C.; Liu, Q.; Fournier-Viger, P.; Djenouri, Y. A(k, p)-anonymity framework to sanitize transactional database with personalized sensitivity. *J. Internet Technol.* **2019**, *20*, 801–808.
6. Kacha, L.; Zitouni, A.; Djoudi, M. KAB: A new k-anonymity approach based on black hole algorithm. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 4075–4088. [CrossRef]
7. Dwork, C. Differential Privacy. In *Automata, Languages and Programming. ICALP 2006*; Lecture Notes in Computer Science; Bugliesi, M., Preneel, B., Sassone, V., Wegener, I., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4052. [CrossRef]
8. Dwork, C.; Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.* **2013**, *9*, 211–407. [CrossRef]
9. Geng, Q.; Viswanath, P. The optimal noise-adding mechanism in differential privacy. *IEEE Trans. Inf. Theory* **2016**, *62*, 925–951. [CrossRef]
10. He, J.; Cai, L.; Guan, X. Differential private noise adding mechanism and its application on consensus algorithm. *IEEE Trans. Signal Process.* **2020**, *68*, 4069–4082. [CrossRef]
11. Sarkar, A.; Sharma, A.; Gill, A.; Thakur, P. A differential privacy-based system for efficiently protecting data privacy. In Proceedings of the 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), Coimbatore, India, 14–16 June 2023; pp. 1399–1404. [CrossRef]
12. Jain, P.; Gyanchandani, M.; Khare, N. Differential privacy: Its technological prescriptive using big data. *J. Big Data* **2018**, *5*, 15. [CrossRef]
13. Farias, V.A.; Brito, F.T.; Flynn, C.; Machado, J.C.; Majumdar, S.; Srivastava, D. Local dampening: Differential privacy for non-numeric queries via local sensitivity. *VLDB J.* **2023**, *32*, 1191–1214. [CrossRef]
14. Cassandras, C.G.; Lafortune, S. Systems and Models. In *Introduction to Discrete Event Systems*; Springer: Cham, Switzerland, 2021; pp. 1–52. [CrossRef]
15. Lin, F. Opacity of discrete event systems and its applications. *Automatica* **2011**, *47*, 496–503. [CrossRef]
16. Badouel, E.; Bednarczyk, M.A.; Borzyszkowski, A.M.; Caillaud, B.; Darondeau, P. Concurrent secrets. *Discrete Event Dyn. Syst.* **2007**, *17*, 425–446. [CrossRef]
17. Zhang, K. State-based opacity of real-time automata. In Proceedings of the 27th IFIP WG 1.5 International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2021), Marseille, France, 12–14 July 2021; Castillo-Ramirez, A., Guillon, P., Perrot, K., Eds.; Volume 90, pp. 12:1–12:15. [CrossRef]
18. Lai, A.; Lahaye, S.; Li, Z. Initial-state detectability and initial-state opacity of unambiguous weighted automata. *Automatica* **2021**, *127*, 109490. [CrossRef]
19. Han, X.; Zhang, K.; Zhang, J.; Li, Z.; Chen, Z. Strong current-state and initial-state opacity of discrete-event systems. *Automatica* **2023**, *148*, 110756. [CrossRef]
20. Balun, J.; Masopust, T. On verification of weak and strong k-step opacity for discrete-event systems. *IFAC-PapersOnLine* **2022**, *55*, 108–113. [CrossRef]
21. Yin, X.; Li, Z.; Wang, W.; Li, S. Infinite-step opacity and k-step opacity of stochastic discrete-event systems. *Automatica* **2019**, *99*, 266–274. [CrossRef]
22. Balun, J.; Masopust, T. On opacity verification for discrete-event systems. *IFAC-PapersOnLine* **2020**, *53*, 2075–2080. [CrossRef]
23. Jones, A.; Leahy, K.; Hale, M. Towards differential privacy for symbolic systems. In Proceedings of the 2019 American Control Conference (ACC), Philadelphia, PA, USA, 10–12 July 2019; pp. 372–377. [CrossRef]
24. Saboori, A.; Hadjicostis, C.N. Verification of initial-state opacity in security applications of DES. In Proceedings of the 2008 9th International Workshop on Discrete Event Systems, Gothenburg, Sweden, 28–30 May 2008; pp. 328–333. [CrossRef]
25. Keroglou, C.; Hadjicostis, C.N. Initial state opacity in stochastic DES. In Proceedings of the 2013 IEEE 18th Conf. Emerging Technol. and Factory Autom. (ETFA), Cagliari, Italy, 10–13 September 2013; pp. 1–8. [CrossRef]
26. Basile, F.; De Tommasi, G.; Motta, C.; Sterle, C. Necessary and sufficient condition to assess initial-state-opacity in live bounded and reversible discrete event systems. *IEEE Control Syst. Lett.* **2022**, *6*, 2683–2688. [CrossRef]
27. Tong, Y.; Li, Z.; Seatzu, C.; Giua, A. Verification of state-based opacity using Petri nets. *IEEE Trans. Automat. Contr.* **2017**, *62*, 2823–2837. [CrossRef]
28. Cong, X.; Fanti, M.P.; Mangini, A.M.; Li, Z. On-line verification of initial-state opacity by Petri nets and integer linear programming. *ISA Trans.* **2019**, *93*, 108–114. [CrossRef] [PubMed]
29. Zhang, K.; Yin, X.; Zamani, M. Opacity of nondeterministic transition systems: A (bi)simulation relation approach. *IEEE Trans. Automat. Contr.* **2019**, *64*, 5116–5123. [CrossRef]

30. Hadjicostis, C.N.; Keroglou, C. Opacity formulations and verification in discrete event systems. In Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), Barcelona, Spain, 16–19 September 2014; pp. 1–12. [CrossRef]
31. Teng, Y.; Li, Z.; Yin, L.; Wu, N. State-based differential privacy verification and enforcement for probabilistic automata. *Mathematics* **2023**, *11*, 1853. [CrossRef]
32. Steinke, T. Composition of differential privacy and privacy amplification by subsampling. *arXiv* **2022**, arXiv:2210.00597.
33. Cassandras, C.G.; Lafortune, S. Languages and automata. In *Introduction to Discrete Event Systems*; Springer: Cham, Switzerland, 2021. [CrossRef]
34. Kumar, R.; Garg, V. Control of stochastic discrete event systems: Synthesis. In Proceedings of the IEEE Conference on Decision and Control, Tampa, FL, USA, 18 December 1998; Volume 3, pp. 3299–3304. [CrossRef]
35. Rabiner, L.R. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* **1989**, *77*, 257–286. [CrossRef]
36. McSherry, F.; Talwar, K. Mechanism design via differential privacy. In Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), Providence, RI, USA, 21–23 October 2007; pp. 94–103. [CrossRef]
37. Jacob, R.; Lesage, J.-J.; Faure, J.-M. Overview of discrete event systems opacity: Models, validation, and quantification. *Annu. Rev. Control* **2016**, *41*, 135–146. Available online: https://www.sciencedirect.com/science/article/pii/S1367578816300189 (accessed on 13 July 2023). [CrossRef]