Differential Privacy for Symbolic Systems with Application to Markov Chains

Bo Chen\textsuperscript{a}, Kevin Leahy\textsuperscript{b}, Austin Jones\textsuperscript{c}, Matthew Hale\textsuperscript{a}

\textsuperscript{a}Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL, USA.

\textsuperscript{b}Massachusetts Institute of Technology Lincoln Laboratory, Lexington, MA, USA.

\textsuperscript{c}No affiliation. Contributions made while this author was affiliated with MIT Lincoln Laboratory, Lexington, MA, USA.

Abstract

Data-driven systems are gathering increasing amounts of data from users, and sensitive user data requires privacy protections. In some cases, the data gathered is non-numerical or symbolic, and conventional approaches to privacy, e.g., adding noise, do not apply, though such systems still require privacy protections. Accordingly, we present a novel differential privacy framework for protecting trajectories generated by symbolic systems. These trajectories can be represented as words or strings over a finite alphabet. We develop new differential privacy mechanisms that approximate a sensitive word using a random word that is likely to be near it. An offline mechanism is implemented efficiently using a Modified Hamming Distance Automaton to generate whole privatized output words over a finite time horizon. Then, an online mechanism is implemented by taking in a sensitive symbol and generating a randomized output symbol at each timestep. This work is extended to Markov chains to generate differentially private state sequences that a given Markov chain could have produced. Statistical accuracy bounds are developed to quantify the accuracy of these mechanisms, and numerical results validate the accuracy of these techniques for strings of English words.

Key words: Differential Privacy; Symbolic Systems; Markov Chains.

1 Introduction

As control applications have become increasingly reliant on user data, there has arisen interest in protecting individuals' privacy, e.g., in smart power grids [1,2] and smart transportation systems [3,4]. Researchers have proposed various quantitative definitions of privacy, and the notion of differential privacy has emerged as one standard privacy specification in recent years [5,6].

The statistical nature of differential privacy makes it unlikely for an eavesdropper or adversary to learn anything meaningful about sensitive data from its differentially private form. Its key features include immunity to post-processing [7], in that transformations of privatized data do not weaken privacy guarantees, and robustness to side information, in that learning additional information about data-producing entities does not substantially weaken differential privacy [8]. Immunity to post-processing means that differentially private data can be freely used without harming privacy’s guarantees. And there is a growing body of work on differential privacy in systems and control that exploits this property, including in multi-agent control [5,9,10], convex optimization [11,12,13,14], linear-quadratic control [15,16], controller design [17], and filtering and estimation problems [18,19]. These works implement differential privacy for numerical data using the Laplace or Gaussian mechanisms, which add noise to sensitive data before sharing it.

Symbolic control systems generate sequences of non-numerical data, which can often be represented as words or strings over a finite alphabet. A symbolic trajectory can represent, for example, a sequence of modes to switch...
between in a hybrid system [20,21], a sequence of regions of state space to occupy, e.g., in a path planning problem [22], or the statuses of smart devices in the Internet of Things [23,24]. In these and other applications, symbolic systems require privacy protections just as their numerical counterparts do. For example, the locations of a patrolling robot over time may require protections in order to prevent an adversary from predicting where the robot will go next. In addition, symbolic trajectories can be produced by Markov chains and Markov decision processes, which have been used to model traffic systems [25], smart buildings [26], cyber physical systems [27], and robot navigation systems [28]. Symbolic trajectories produced by these systems can reveal the locations of a user, their activities, and their acquaintances, yet it is desirable to share such trajectories to enable multi-agent coordination, learning from trajectory data, and other data-driven applications. Therefore, we develop novel differential privacy protections that enable privatized symbolic data to be shared, and our developments apply to these applications and any others in which symbolic data contains sensitive information, including any applications modeled as Markov chains.

Non-numerical data typically cannot be privatized with additive noise, but it can be kept differentially private using the exponential mechanism [7]. The exponential mechanism takes non-numerical data as input and randomly outputs non-numerical data based on its “quality”, which is user-specified. There is not an inherent notion of quality for symbolic systems, and in this work we choose to use the Hamming distance as the notion of quality, which means that a private word is of high quality if it is close to the sensitive word it approximates. It is known that the exponential mechanism can have high computational complexity when its input and output spaces are large [7]. Indeed, Section 3.4 shows that the computational complexity of a naïve implementation for strings would be exponential in the length of those strings. Thus, more tractable approach is required.

Accordingly, this paper develops an efficient method for the privatization of sensitive words generated by symbolic systems. The first contribution of this paper is the definition of differential privacy itself in this context. Given a sensitive input word, differential privacy requires the generation of private output words that are near the input word (in the Hamming sense) with high probability. The second contribution is a computationally efficient privacy mechanism that constructs and uses a modified Hamming Distance nondeterministic finite state automaton (NFA) to generate private output words. This mechanism operates on whole strings offline.

In some applications, real-time reporting of status information is necessary for effective operation, e.g., real-time IoT monitoring systems [29] and smart home management systems [30]. Also, in cloud control [31], states are transmitted to an aggregator and the aggregator sends back commands in real-time. To provide privacy in such applications, the third contribution of this work is an online differential privacy mechanism. It generates individual random symbols in a way that privatizes entire symbolic trajectories. This mechanism has the advantage that future symbolic states are not needed a priori.

The fourth contribution is the extension of these differential privacy mechanisms to Markov chains. Markov processes specifically have been considered in applications such as traffic systems [32] and healthcare [33]. These system setups require the sharing of symbolic trajectories that can be sensitive because they may contain a user’s destination or personal health information. Each sensitive symbol in a Markov chain trajectory is dependent on its previous symbol, and only some symbol-to-symbol transitions are feasible. Thus, both the offline and online mechanisms are modified to generate output trajectories that are feasible with respect to the dynamics of a given Markov chain.

Concentration bounds are developed for the accuracy of each mechanism, which enable the calibration of privacy protections based on the acceptable error in a given application. These mechanisms are demonstrated on a Markov chain generated by the traffic data for some of the major streets in Gainesville, Florida, which is available at Florida Traffic Online (2021) [34].

A preliminary version of this work appeared in [35]. The current paper extends this work in three ways. First, we provide a full proof that the mechanism in [35] provides differential privacy, and we also derive novel error bounds for it. Second, we provide a new differential privacy mechanism for the online setting, which can be implemented for real-time control, and accuracy bounds are provided for it as well. Third, both the offline and online mechanisms are extended to Markov chains and two new privacy mechanisms are developed. Both are shown to provide differential privacy while ensuring that all privately generated words are feasible with respect to the dynamics of the underlying Markov chain. Error bounds are also presented for this mechanism.

Other approaches to protecting information include opacity [36], which bounds the probability of correct state estimates by an adversary that observes all actions in an MDP [37,38]. Our paper differs by considering problems in which a user chooses to share information privately, rather than maintaining secrecy under observation.

A related body of work introduced approximate opacity [39], which studies the protection of the states of a system while an intruder makes approximate observations of that system. This work differs by considering a setting in which differentially private state observations are deliberately shared with an outside party,
rather than being the product of observations of a system. That is, while approximate opacity studies inherent system properties that make it difficult to infer the system’s states, we develop new private output maps that can be added to an existing system to enforce differential privacy. Additionally, in [40,41], policies are synthesized to minimize the predictability of trajectories to an outside observer by maximizing the entropy of MDPs. Conversely, we develop private output maps that do not require re-synthesizing a policy for privacy. In addition, our approaches enforce differential privacy at the trajectory level, rather than pointwise in time [42].

The rest of the paper is organized as follows. Section 2 presents background, and Section 3 gives formal problem statements. Section 4 develops privacy mechanisms, and Section 5 applies them to Markov chains. Section 6 gives numerical results, and Section 7 concludes.

**Notation** Let $\mathbb{N}$ denote the set of all non-negative integers and $\mathbb{N}^+$ denote the set of all positive integers. For $n \in \mathbb{N}^+$, let $[n] = \{1, \ldots, n\}$, $\Sigma$ denotes a finite alphabet. A word of length $n$ over $\Sigma$ is a concatenation of symbols $w = \sigma_1\sigma_2\ldots\sigma_n$ with $\sigma_i \in \Sigma$ for all $i \in [n]$. We also write $w_i$ for the $i^{th}$ symbol in the word $w$. Let $\Sigma^n$ denote all words of length $n$ over $\Sigma$, and $2^\Sigma$ denote the power set of $\Sigma$. The notation $|w|$ denotes the length of a word $w$.

## 2 Preliminaries on Symbolic Systems

A finite state automaton (FSA) is a tuple $A = (Q, \Sigma, q^0, \delta, F)$, where $Q$ is a set of states, $\Sigma$ is an input alphabet, $q^0 \in Q$ is the initial state, $\delta : Q \times \Sigma \rightarrow Q$ is the transition function between states, and $F \subseteq Q$ is the set of accepting states. If the transition function $\delta$ is a nondeterministic mapping, i.e. $\delta : Q \times \Sigma \rightarrow 2^Q$, then this FSA is called a nondeterministic finite state automaton (NFA).

Given an NFA $A = (Q, \Sigma, q^0, \delta, F)$, a word $w = \sigma_1\sigma_2\ldots\sigma_n$, with $\sigma_i \in \Sigma$, induces a run, which is a word $q_0q_1\ldots q_n \in Q^\ast$ such that $q_0 = q^0$ and $q_{i+1} \in \delta(q_i, \sigma_{i+1})$. The automaton $A$ accepts a word $w_n$ if the final state of the induced run is an accepting state, i.e., $q_n \in F$. The set of words accepted by $A$ is its language, denoted $L(A)$.

To compare two words, we introduce the Hamming distance.

**Definition 1 (Hamming Distance)** Given an alphabet $\Sigma$, for two $n$-length words $v, w \in \Sigma^n$, the Hamming distance between them, denoted $d(v, w)$, is the number of positions at which the corresponding symbols are different. Mathematically, we have $d(v, w) = |\{i \mid v_i \neq w_i\}|$, where $| \cdot |$ denotes cardinality.

In other words, the Hamming distance is a metric that measures the minimum number of substitutions that can be applied to $v$ to convert it to $w$. For example, the Hamming distance between “hammer” and “bumper” is 3.

## 3 Privacy Background and Problem Statements

This section gives background and the problem statements that are the focus of the remainder of the paper.

### 3.1 Basic Differential Privacy Definitions

Differential privacy is enforced by a mechanism, which is a randomized map. For similar pieces of sensitive data, a mechanism must produce outputs that are approximately distinguishable. The definition of “similar” is given by an adjacency relation, which takes the following form for words generated by a symbolic system.

**Definition 2 (Word Adjacency)** Fix a length $n \in \mathbb{N}^+$ and an adjacency parameter $k \in \mathbb{N}$. The word adjacency relation on $\Sigma^n$ is $\text{Adj}_{n,k} = \{(w_1, w_2) \mid d(w_1, w_2) \leq k\}$.

Two words in $\Sigma^n$ are adjacent if the Hamming distance between them is no more than $k$. For example, consider Figure 1, where the robot can travel from $(1, 1)$ to $(5, 5)$ along Path 1, 2, or 3. Each path gives a length 9 word of grid cells the robot traverses. With $k = 3$, Paths 1 and 2 are word adjacent (since they differ in two points), but Path 3 is not word adjacent to the other two (due to differing in more than 3 points). Adjacent words must be made approximately indistinguishable, and $k$ is the size of difference that must be masked.

We define adjacency in terms of the Hamming distance because this allows the mechanisms we develop to protect sensitive differences between trajectories. For example, suppose a vehicle unexpectedly deviates from its nominal commute and that we would like to mask this deviation using differential privacy. If the nominal trajectory and the deviating trajectory are adjacent to each other, then differential privacy will render the deviating trajectory approximately indistinguishable from the nominal trajectory, thereby concealing the sensitive deviation. It is precisely these protections that we attain by defining adjacency in terms of the Hamming distance between such trajectories. We note as well that differential privacy for state space systems (with numerical trajectories) defines adjacency in terms of an appropriate metric in an $\ell_p$-space [18], and thus our use of the Hamming distance is the natural analog of existing work for the symbolic setting.

We next introduce the definition of differential privacy for symbolic systems we use throughout this paper.
3.2 Differential Privacy Problem Statements

Formal privacy problem statements are given next.

Definition 3 (Word Differential Privacy) Fix a probability space \((\Omega, \mathcal{F}, \mathbb{P})\), an adjacency parameter \(k \in \mathbb{N}\), a length \(n \in \mathbb{N}^+\), and a privacy parameter \(\epsilon > 0\). A mechanism \(M_w : \Sigma^n \times \Omega \rightarrow \Sigma^n\) is word \(\epsilon\)-differentially private if, for all words \((w_1, w_2) \in \text{Adj}_{n,k}\) and all \(L \subseteq \Sigma^n\), it satisfies \(\mathbb{P}[M_w(w_1) \in L] \leq e^\epsilon \mathbb{P}[M_w(w_2) \in L]\).

The parameter \(\epsilon\) sets the strength of privacy protections, and smaller \(\epsilon\) implies stronger privacy. In the literature, \(\epsilon\) typically ranges from 0.01 to 10 [43]. A word differential privacy mechanism guarantees that the randomized outputs of two \(k\)-adjacent words will be made approximately indistinguishable to any recipient of their privatized forms, including any eavesdroppers. Thus, these recipients are unlikely to determine the underlying sensitive word or make high-confidence inferences about it.

We note that other metrics can be defined on the set of words that allow for the comparison of words of different lengths. One example is the Levenshtein distance [44], which allows substitutions of symbols between words, as well as insertions and deletions. However, we do not use such a metric for adjacency because doing so would allow a privacy mechanism to generate private outputs whose lengths are different from the lengths of the inputs that produce them. This can be problematic, particularly when a private output is shorter than the sensitive input that produced it. To see why, consider a symbolic trajectory that captures the sequence of intersections traversed by an autonomous vehicle. If a privacy mechanism deletes entries of such a trajectory, then it has deleted data about the location history of that vehicle. In doing so, the privacy mechanism can generate semantically invalid data, e.g., a sequence of intersections that cannot actually be consecutively followed from the starting point to the ending point. Such instances motivate us to use the Hamming distance in this work because it can only allow words to be adjacent if they have the same length.

Problem 1 Fix a probability space \((\Omega, \mathcal{F}, \mathbb{P})\). Given a word length \(n \in \mathbb{N}^+\), an alphabet \(\Sigma\), an adjacency parameter \(k \in \mathbb{N}\), and an adjacency relation \(\text{Adj}_{n,k}\), develop an offline word \(\epsilon\)-differential privacy mechanism \(M_w^{\text{off}} : \Sigma^n \times \Omega \rightarrow \Sigma^n\) that takes as input an entire word \(w\) that is already generated, i.e., for all \(i \in [n]\), \(\sigma_i\) is known a priori.

Problem 1 mathematically formulates an offline mechanism, in the sense that it privatizes an entire word after it has been generated. This would be used when data is first harvested, then privatized and released in batches.

We are also interested in the online setting to account for cases in which a symbolic trajectory must be privatized and shared as it is generated. One symbol is shared at each point in time, and thus \(n\) is both the length of a word and the length of time horizon over which it is shared. For a word \(w = \sigma_1 \sigma_2 \ldots \sigma_n\), the online setting shares \(\sigma_t\) for each time \(t \in [n]\), which leads to the following.

Problem 2 Fix a probability space \((\Omega, \mathcal{F}, \mathbb{P})\). Given a word length \(n \in \mathbb{N}^+\), an alphabet \(\Sigma\), an adjacency parameter \(k \in \mathbb{N}\), and an adjacency relation \(\text{Adj}_{n,k}\), develop an online mechanism \(M_w^{\text{on}} : \Sigma^n \times \Omega \rightarrow \Sigma^n\) that is word \(\epsilon\)-differentially private, where only the symbols \(\sigma_1 \ldots \sigma_t\) of \(w\) are known at each time \(t\), i.e., the future symbols \(\sigma_{t+1} \ldots \sigma_n\) after time \(t\) are unknown.

For a sensitive word \(w = \sigma_1 \sigma_2 \ldots \sigma_n \in \Sigma^n\), the online mechanism \(M_w^{\text{on}}\) approximates the sensitive symbol \(\sigma_t\) at each \(t \in [n]\) with a random symbol. The challenge is that, for any time \(t \in [n]\), \(M_w^{\text{on}}\) only has access to symbols generated before time \(t\), while differential privacy must be enforced for the entire word. Thus, the symbol-by-symbol randomization of \(M_w^{\text{on}}\) must enforce the correct distribution over entire words without knowing the entire words. Problems 1 and 2 are solved in Section 4.

3.3 Extension to Markov Chains

A widely used class of symbolic system models is Markov chains. A Markov chain is a sequence of random variables \(X_1, X_2, \ldots \in S\) with the Markov property, i.e., the value of \(X_{t+1}\) depends only on the value of \(X_t\). The state space \(S\) contains all possible values of \(X_t\). The transition probability of going from state \(s_i\) to state \(s_j\) in one step is \(p_{i,j} = \mathbb{P}[s_j | s_i]\). The matrix of transition probabilities is denoted \(P\), where \(P_{ij} = p_{i,j}\). In this work, the tuple \((S, P, s_0)\) denotes the Markov chain with state space \(S\), transition matrix \(P\), and initial state \(s_0 \in S\). A state \(s_j\) is called a feasible state of another state \(s_i\) if \(\mathbb{P}[s_j | s_i] > 0\). For a Markov chain with state space \(S\), let \(S^*\) denote all sequences it can generate in finite time and \(S^n\) denote all sequences that is in length \(n\). Any such sequence over a finite horizon \([n]\) can be identified with a word \(w = s_0 s_1 \ldots s_n \in S^n\). If for all \(t \in [n-1]\), \(\mathbb{P}[s_{t+1} | s_t] > 0\), i.e., every state is a feasible state of its
previous state, then the word $w$ is called feasible. The set of feasible words of length $n$ is denoted $L(S^n)$.

For privacy of state sequences of Markov chains, the goal is to generate a private sequence of states which is feasible with respect to its dynamics.

**Problem 3** Fix a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Given a word length $n \in \mathbb{N}^+$, an alphabet $\Sigma$, an adjacency parameter $k \in \mathbb{N}$, and an adjacency relation $\text{Adj}_{n,k}$, for a Markov chain $(S, P, s_0)$, develop word $\epsilon$-differentially private offline and online mechanisms, $M^w_{\text{off}}$ and $M^w_{\text{on}}$, such that each output word is feasible with respect to the allowable transitions of the Markov chain. The offline mechanism takes in an entire word $w$, while the online mechanism only has access to the symbols $\sigma_1 \cdots \sigma_t$ at each time $t$, i.e., the future symbols $\sigma_{t+1} \cdots \sigma_n$ after time $t$ are unknown.

We note here that Problems 1, 2, and 3 merely state the privacy requirements of each setting, but they deliberately do not constrain the allowable implementations of the privacy mechanisms that solve them. For example, in Problem 2, a privacy mechanism may have some form of internal memory so that each output symbol depends on the input string and past output symbols. This setup is permitted despite not being a requirement in Problem 2. Indeed, a solution to each problem requires only some mechanism that enforces differential privacy and maps between the appropriate spaces, regardless of its internal implementation details.

Our study of Markov chains is motivated by our interest in the development of trajectory-level privacy for symbolic systems, analogous to the notion of trajectory-level privacy developed for state space systems in [18]. In that work, stochastic systems are studied, which means that privacy at the trajectory level does not collapse to privacy of the initial state of the system (as would be the case for deterministic systems, e.g., [45,46,47,48,49]). Similarly, in this work we do not consider deterministic finite-state automata because their determinism does cause such a collapse of privacy. Instead, we consider Markov chains because their stochastic dynamics make it meaningful to define privacy at the level of symbolic trajectories.

### 3.4 Background: Exponential Mechanism

This section briefly reviews the exponential mechanism, which is the foundation for the privacy mechanisms we develop (Section 3.4 of [7] gives background). We emphasize that this paper does not merely implement the exponential mechanism. Instead, as described in Remark 2 below, the exponential mechanism would be computationally prohibitive to use in its standard form, and this work must develop significantly less computationally complex mechanisms. This subsection simply gives the privacy goals that motivate this work, and actually attaining those goals will be the focus of this work.

For a non-numerical query $f$ with range $\mathcal{R}$, the exponential mechanism is implemented by first computing the query $f$ on a given input $w_i$, and then sampling a random output from $\mathcal{R}$ that suitably approximates $f(w_i)$. Mathematically, the probability of selecting an output depends on its utility score, and outputs with high utility scores are generated with high probability. In this work, the query under consideration is the identity query, i.e., $f(w_i) = w_i$, which means that we must randomize an entire symbolic word, rather than some function of it. The definition of utility score is a design choice made in developing an exponential mechanism, and, given the use of the identity query, this work uses the following.

**Definition 4 (Utility function)** Fix an alphabet $\Sigma$ and a length $n \in \mathbb{N}^+$, and consider a sensitive input word $w_i \in \Sigma^n$. Then a private output word $w_o \in \Sigma^n$ provides utility equal to $u(w_i, w_o) = -d(w_i, w_o)$.

This utility function encodes the fact that $w_o$ is a better output for $w_i$ when it is close to $w_i$. The probability distribution used by the exponential mechanism depends on both the value of $u$ and its sensitivity, defined next.

**Lemma 1 (Sensitivity [35])** Fix an alphabet $\Sigma$, a length $n \in \mathbb{N}^+$, a set $L \subset \Sigma^n$, and an adjacency parameter $k \in \mathbb{N}$. Then the sensitivity of $u$ is

$$\Delta u = \max_{v \in L} \max_{w_1, w_2 \in L} \max_{(w_1, w_2) \in \text{Adj}_{n,k}} |u(w_1, v) - u(w_2, v)| \leq k.$$ 

The sensitivity bounds the amount by which $u$ can differ between two adjacent input words in $L$. One can have $L = \Sigma^n$, though Lemma 1 also allows for $L \subset \Sigma^n$ for systems in which some words are infeasible.

**Definition 5 (Exponential Mechanism)** Fix an alphabet $\Sigma$, a length $n \in \mathbb{N}^+$, a language $L \subseteq \Sigma^n$, and an adjacency parameter $k \in \mathbb{N}$. For a sensitive input word $w_i \in L$, the exponential mechanism $M_e$ outputs $w_o \in L$ with probability

$$p_e(w_o) = \frac{\exp\left(\frac{\epsilon u(w_i, w_o)}{2\Delta u}\right)}{\sum_{w \in L} \exp\left(\frac{\epsilon u(w_i, w)}{2\Delta u}\right)} = K_w \exp\left(\frac{\epsilon u(w_i, w_o)}{2\Delta u}\right).$$

**Remark 1** For two private output words $w_1$ and $w_2$ and an input word $w_1$, if $d(w_1, w_1) = d(w_1, w_2)$, then then exponential mechanism will output them with equal probability. This is an essential property that we will use to design efficient privacy mechanisms in later sections.
Remark 2 As stated, this would provide word \( \epsilon \)-differential privacy, though directly implementing the exponential mechanism over \( \Sigma^n \) is infeasible. For a fixed \( w_1 \), to determine the proportionality constant \( K_w \), one would need to compute the Hamming distance from \( w_1 \) to every possible output word. There are \( m^n \) total strings of length \( n \) on an alphabet of \( m \) symbols, and the time complexity of computing the required distances is \( O(mn^2) \), which is prohibitive, especially for long strings and large alphabets. Therefore, there is a need to develop new, efficient mechanisms, which is done next.

4 Differential privacy over a finite alphabet

This section solves Problems 1 and 2 in Subsections 4.1 and 4.2, respectively.

4.1 Offline Mechanism

To develop an efficient offline privacy mechanism, we introduce an automaton, called the modified Hamming distance NFA, which identifies all words that have a specific Hamming distance to an input word.

Definition 6 (Modified Hamming Distance NFA) Fix an alphabet \( \Sigma \) and a length \( n \in \mathbb{N}^+ \). For a word \( x \in \Sigma^n \) and a distance \( j \in \mathbb{N} \), the modified Hamming Distance NFA (MNFA) is an NFA \( A_{x,j} = (Q_{x,j}, \Sigma, q_0, \delta, F_{x,j}) \) such that \( L(A_{x,j}) \) is the set of all words of length \( n \) with Hamming distance from \( x \) equal to \( j \). Each state \( q \in Q_{x,j} \) can transfer to another state by a policy \( \mu(\cdot, q, \sigma) \). Formally, we have

\[
\forall q_i = q_{i+1}, \sigma_{i+1} \text{ s.t. } q_i \in Q_{x,j}, \quad \mu(q_{i+1}, \sigma_{i+1} | q_i) = 1,
\]

where \( \mu(q_{i+1}, \sigma_{i+1} | q_i) \) is the probability that the input symbol \( \sigma_{i+1} \) causes a transition from \( q_i \) to \( q_{i+1} \).

Algorithm 1 takes as input a sensitive string \( w_1 \), the transition function \( \delta \), and the accepting set \( \{q_{n,\ell}\} \). It outputs a policy \( \mu_{w_1} \), which is designed as follows. We assign a function \( V : Q_{w_1,\ell} \to \mathbb{N} \) such that \( V(q) \) is the number of unique paths from the state \( q \in Q_{w_1,\ell} \) that end in \( \{q_{n,\ell}\} \). The probability of outputting a symbol \( \sigma \in \Sigma \) at each state \( q \) is equal to the proportion of \( V(q) \) compared to \( V(\delta(q, \sigma)) \). This ensures that all potential private outputs words distance \( \ell \) from \( w_1 \) are equiprobable.
Fig. 2. Algorithm 1 for all words of length 3 and distance 2 from the word $abc$ over the alphabet $\{a, b, c\}$. Each circle represents a state $q \in Q_{w_i, t}$, and each arrow represents a feasible transition. The state with a double circle denotes the accepting state $q_{n, t}$. The value of $V(q)$ is in blue. The value of $\mu_{w_i}$ is in red with the output letter in front of it.

Algorithm 1 is backward reachable: it starts at an accepting state $q_{n, t}$ and sets $V(q_{n, t}) = 1$ and $\text{CurrQ} = \{q_{n, t}\}$. It then loops through all states that can reach any state in $\text{CurrQ}$, and finds the corresponding unique paths. These states are stored in the set $\text{ActiveQ}$. At the end of each iteration, we set $\text{ActiveQ}$ to be the new $\text{CurrQ}$ set, and we reinitialize $\text{ActiveQ}$. Then this process repeats until it reaches the initial state.

In Equation (1), $p(\ell; w_i, k)$ induces a probability distribution over output words, and Algorithm 1 provides the means to efficiently sample from it. An example output of Algorithm 1 is shown in Figure 2.

**Theorem 1 (Solution to Problem 1)** Given an adjacency parameter $k \in \mathbb{N}$, a privacy parameter $\epsilon \geq 0$, and a sensitive word $w_i \in \Sigma^n$, Mechanism 1 provides word $\epsilon$-differentially privacy to $w_i$ with respect to the adjacency relation $\text{ Adj}_{n, k}$ in Definition 2.

Proof: See Appendix A.

For a Hamming distance $\ell$ generated with Equation (1), the time complexity for an alphabet that contains $m$ symbols to generate a private output using Mechanism 1 is $O(nm)$, which is a significant improvement over the direct implementation of the exponential mechanism (cf. Remark 2). This is possible because Algorithm 1 narrows down the set of possible private outputs significantly by restricting the potential outputs to those with distance exactly $\ell$ from $w_i$.

There is also a need to analyze the accuracy of private words. Such analyses enable informed calibration of privacy by balancing the strength of privacy protections with the errors they induce. This is done by computing the expectation and variance of the distance between input and output words as a function of $\epsilon$.

**Theorem 2** Fix an alphabet $\Sigma$ that contains $m$ symbols, a privacy parameter $\epsilon \geq 0$, an adjacency parameter $k \in \mathbb{N}$, and an input word $w_i \in \Sigma^n$. Under Mechanism 1, the distance $\ell$ to the private output satisfies

$$E[\ell] = n - \frac{n}{(m - 1) \exp(-\frac{\epsilon}{2k}) + 1}.$$ 

$$\text{Var}[\ell] = \frac{n[(m - 1) \exp(-\frac{\epsilon}{2k})] + 1)^2}{(m - 1) \exp(-\frac{\epsilon}{2k}) + 1}.$$ 

For any $\eta \in (0, 0.5)$, we have the concentration bound $\mathbb{P}[|d(w_i, w_o) - E[d(w_i, w_o)]| > \eta] \leq 2 \exp \left( - \frac{2\eta^2}{n} \right)$.

Proof: See Appendix B.

**Remark 3** When $\epsilon \to 0$, we see that the expected value $E[\ell] \to n - \frac{n}{m}$ and $\text{Var}[\ell] \to \frac{n(m-1)}{m^2}$. Then highly private outputs are random strings that differ from their inputs in nearly every entry. On the other hand, as $\epsilon \to \infty$, $E[\ell] = 0$ and $\text{Var}[\ell] = 0$. Thus, as privacy vanishes, the mechanism simply outputs the input word unchanged.

### 4.2 Online Mechanism

Future states are unknown in the online setting, and we propose an alternative mechanism: each output character is generated only based on the most recently generated sensitive symbol in a word. The correct output character (correct transition, CT) is assigned a probability $\mu(c_T | s_T)$, and other symbols (substitutions, ST) are set to be equiprobable.

**Mechanism 2 (Online Mechanism)** Fix a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Given an alphabet $\Sigma$ that contains $m$ symbols and a word $w_i = \sigma_1 \ldots \sigma_n \in \Sigma^n$, the online mechanism $M_{w_i}^\text{sn}$ chooses an output word $w_o = \sigma_1^o \ldots \sigma_n^o \in \Sigma^n$ by selecting each $\sigma_i^o$ from the distribution $\mathbb{P}[\sigma_i^o] = \mu(c_T | \sigma_t) = \mu_T(c_T | \sigma_T)$, where $\mu_T(c_T | \sigma_T) : \Sigma \to [0, 1]$ is a policy generated by Algorithm 2.

**Algorithm 2:** Policy construction for online setting (Implements Mechanism 2; Solves to )

**Input:** Sensitive word $w_i$, probability of correct transition $\tau$

**Output:** Policy $\mu_T$

1. for $\sigma_1 \in \Sigma$ do
2. for $\sigma_1^o \in \Sigma$ do
3. if $\sigma_1^o = \sigma_1$ then
4. $\mu_T(\sigma_1^o | \sigma_1) = \tau$; // CT
5. else
6. $\mu_T(\sigma_1^o | \sigma_1) = \frac{1-\tau}{m-1}$; // ST
7. end
8. end
9. end

At each time $t$, the online mechanism $M_{w_i}^\text{sn}$ calls the policy $\mu_T$, which takes $\sigma_t$ as input and outputs a probability
distribution over $\Sigma$ in terms of $\tau$. The value of $\tau$ must be chosen to enforce differential privacy.

**Theorem 3 (Solution to Problem 2)** Fix an alphabet $\Sigma$ that contains $m$ symbols, a sensitive word $w_t \in \Sigma^n$, a privacy parameter $\epsilon \geq 0$, and an adjacency parameter $k \in \mathbb{N}$. Then the online mechanism $M_{\Sigma,0}^n$ is word-$\epsilon$-differentially private with respect to the adjacency relationship $\text{Adj}_{0,n}$ in Definition 2 if $\tau$ in Algorithm 2 satisfies

$$\tau = \frac{1}{(m-1) \exp(-\frac{\epsilon}{k}) + 1}.$$

**Proof:** See Appendix C. $\blacksquare$

For a privacy level $\epsilon$, one only needs to construct the policy $\mu$ once with time complexity $O(m^2)$. Then every time the online mechanism receives an input symbol, one can call the policy $\mu_e$ to generate a private output symbol. The accuracy of Mechanism 2 is quantified next.

**Theorem 4** Fix an alphabet $\Sigma$ that contains $m$ symbols, a privacy parameter $\epsilon \geq 0$, and an adjacency parameter $k \in \mathbb{N}$. Suppose Mechanism 2 takes $w_t \in \Sigma^n$ as input and generates $w_o \in \Sigma^n$ as output. Then

$$E[d(w_t, w_o)] = n - \frac{n}{(m-1) \exp(-\frac{\epsilon}{k}) + 1},$$

$$\text{Var}[d(w_t, w_o)] = \frac{n(m-1) \exp(-\frac{\epsilon}{k})}{[(m-1) \exp(-\frac{\epsilon}{k}) + 1]^2}.$$

For $\eta \in (0, 1)$, we have

$$\mathbb{P}[d(w_t, w_o) > (1 + \eta)E[d(w_t, w_o)]] \leq e^{-\frac{n^2 \tau^2}{2}E[d(w_t, w_o)]},$$

$$\mathbb{P}[d(w_t, w_o) < (1 - \eta)E[d(w_t, w_o)]] \leq e^{-\frac{n^2 \tau^2}{2}E[d(w_t, w_o)]}.$$

**Proof:** See Appendix D. $\blacksquare$

**Remark 4** A larger value of $\epsilon$ gives weaker privacy protections. In Theorem 4, both the expectation and variance go to zero as $\epsilon$ grows. Thus, the online mechanism captures the intuition that weaker privacy should give output words closer to the input word.

For a symbolic system whose states take values in $\Sigma$, Mechanism 1 provides the means to privatize state sequences in batches, while Mechanism 2 provides the means to privatize state sequences as they are generated. Both mechanisms generate outputs that take values in the entirety of the state space $\Sigma$, though some systems have restrictions on which state-to-state transitions are feasible. A privacy implementation for such a system must account for these feasibility requirements, and the next section does this for a class of systems.

**5 Generating differentially private runs for a Markov Chain**

This section solves Problem 3. That is, the principles used in Section 4 to generate differentially private words from an arbitrary alphabet are applied to Markov chains. Additional developments are necessary for Markov chains because Mechanisms 1 and 2 must be modified to account for the feasibility of transitions between states.

For a Markov chain with state space $S$, it is in general only possible to transition from a given state $s$ to a subset of other states in $S$. In this section, we define the symbols

$$C(s) = \{s' \in S \mid \mathbb{P}[s' \mid s] > 0\} \text{ and } N(s) = |C(s)|,$$

i.e., $C(s)$ is all states that can be reached from $s$, and $N(s)$ is the number of such states.

**5.1 Offline Mechanism**

The offline mechanism for Markov Chains works similarly to general symbolic systems, except that, to address the feasibility problem, policies are synthesized using a Product modified Hamming distance NFA.

**Definition 7 (Product Modified Hamming Distance NFA)** Let a Markov chain $(S, P, s_0)$ be given. For a sequence of states $x \in S^n$ and a distance $j \in \mathbb{N}$, let $A_{x,j} = (Q_{x,j}, \Sigma, q_0, \delta, F_{x,j})$ be an MNFA. Then the Product Modified Hamming Distance NFA (P-MNFA) is an MNFA $A_{x,j,s} = (Q_{s}, \Sigma, q_0, \delta_s, F_{s})$, where

$$Q_{s} = Q \times S,$$

$$\delta_s : Q \times S \times \Sigma \rightarrow 2^{Q_{s}}, \quad q_0 = (q_0, s_0)$$

and $F_{s} = \{(q_f, s) \in Q_{s} \mid q_f \in F_{n,j}, s \in S\}$, and for any $(q', s') \in \delta_s(q, s, \sigma)$, we have $\delta(q, \sigma) = q'$ and $\mathbb{P}[s' \mid s] > 0$. A state $q_s \in Q_{s}$ can transition to another state by a policy $\mu_s(\cdot \mid q_s, s) : Q_{s} \times S \rightarrow [0, 1]$. $\mathcal{L}(A_{x,j,s})$ is the set of all feasible words of length $n$ with Hamming distance from $x$ equal to $j$.

In words, $A_{x,j,s}$ is the synchronous product of $A_{x,j}$ and the Markov chain $(S, P, s_0)$, and every product transition function $\delta_s$ has to satisfy both the transition function $\delta$ and be a feasible transition in the Markov chain.

**Mechanism 3 (Offline Mechanism for Markov Chains)** Fix a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and a Markov chain $(S, P, s_0)$. Let an adjacency parameter $k \in \mathbb{N}$ be given. Given a sensitive word $w_t = s_1s_2 \cdots s_n \in S^n$, define the offline mechanism $M_{w_t}^{\text{off}}$ (the subscript $s$ indicates “state dependent”), which generates an output word $w_o = s'_1s'_2 \cdots s'_n \in S^n$ by first drawing a Hamming distance from the distribution

$$p(\ell; w_t, k) = \frac{m_\ell \exp\left(-\frac{\ell\delta}{k}\right)}{\sum_{i=0}^{\infty} m_i \exp\left(-\frac{\ell\delta}{k}\right)},$$

(3)
where \( m_\ell \) is the number of feasible words in \( S^n \) with initial state \( s_0 \) that are distance \( \ell \) to the input word. After sampling a distance, it constructs a P-MNFA \( A_{w, k, s} = (Q_{w, k, s}, (q_0, 0, 0, s_0), \delta_s, \{(q_n, \ell, s) \mid s \in S) \}) \) and synthesizes a policy \( \mu_{\epsilon,s} \) using Algorithm 3. An output word \( w_\alpha = s_0^\ell \ldots s_n^\ell \in \mathcal{L}(A_{w, k, s}) \) is generated by running the NFA \( A_{w, k, s} \) once.

Mechanism 3 is similar to Mechanism 1, but with the following differences. First, in Equation (3), \( p(\ell; w_i, k) \) induces a probability distribution over only output words in \( \mathcal{L}(S^n) \). Second, observe that, in Mechanism 3, for every output word \( w_\alpha = s_0^\ell \ldots s_n^\ell \in S^n \), feasibility always holds, i.e., \( \mathbb{P}[s_i^\ell \mid s_i^\ell] > 0 \) for all \( i \in [n - 1] \). This is because in Algorithm 3, each state \( (q, s) \in Q_S \) can transition to the state \( \delta(q, s, s') \) only if \( \mathbb{P}[s' \mid s] > 0 \).

For a given Hamming distance \( d \), the time complexity to generate a feasible private output is \( O(n|L|) \). Figure 3 shows an example of Mechanism 3.

**Algorithm 3:** Product Modified Hamming Distance NFA Construction for Markov Chains

**Input:** Input string \( w_i \), transition function \( \delta_S \), accepting set \( \{(q_n, \ell, s) \mid s \in S) \}

**Output:** policy \( \mu_{\epsilon,s} \)

1. \( n = |w_i| \);
2. \( V_s(q_{|w_i|}, n, s) = 1, \forall s \in S; \)
3. \( \text{CurrQ} = \{(q_{|w_i|}, n, s)\}; \forall s \in S; \)
4. \( \text{ActiveQ} = \{\}; \)
5. \( \text{counter} = 1; \)
6. while \( \text{counter} \leq n \) do
   7. for \((q', s') \in \text{CurrQ} \) do
       8. for \((q, s) \) s.t. \( \delta_S(q, s, s') = (q', s') \) do
           9. \( V_s(q, s) = \frac{1}{\sum_{(q'', s'') \in \text{currQ}} V_s(q'', s'')} V_s(q', s'); \)
           10. \( \mu_{\epsilon,s}(q', s' \mid q, s) = \frac{V_s(q', s')}{V_s(q, s)}; \)
           11. \( \text{ActiveQ} = \text{ActiveQ} + \{(q, s)\}; \)
   12. end
   13. \( \text{CurrQ} = \text{ActiveQ}; \)
   14. \( \text{ActiveQ} = \{\}; \)
   15. \( \text{counter} = \text{counter} + 1; \)
16. end

---

**Theorem 5 (Solution to first part of Problem 3)**

Fix a privacy parameter \( \epsilon \geq 0 \) and an adjacency parameter \( k \in \mathbb{N} \). Suppose the state sequence \( w_i \in S^n \) is generated by the Markov chain \( \mathcal{M} = (S, P, s_0) \). Then Mechanism 3 is \( \epsilon \)-differentially private with respect to the adjacency relation \( \text{Adj}_{n,k} \), and the output word \( w_\alpha \) is feasible for \( \mathcal{M} \).

**Proof:** See Appendix E.

We next bound the error induced by Mechanism 3.

**Theorem 6** Fix a state space \( S \), an adjacency parameter \( k \in \mathbb{N} \), and a privacy parameter \( \epsilon \geq 0 \). De-

---

**Fig. 3.** Mechanism 3 applied to the Markov chain in the upper figure leads to the MNFA in the lower figure.

fine \( N_{\text{max}} = \max_{s \in S} N(s) \) and \( N_{\text{min}} = \min_{s \in S} N(s) \). Then for an input word \( w_i \), the distance between \( w_i \) and a private output word \( w_\alpha \) from Mechanism 3 obeys \( E \leq E[d(w_i, w_\alpha)] \leq E \) and \( \text{Var}[d(w_i, w_\alpha)] \leq \frac{n^2}{4} \).

Using \( B_{r,k} = \exp(-\frac{\epsilon^2}{2k}) \), we have

\[
E[d(w_i, w_\alpha)] = \frac{n(N_{\text{min}} - 1)B_{r,k}[N_{\text{max}} - 1]B_{r,k} + 1]^{n-1}}{\sum_{i=0}^{n} m_i \exp(-\frac{\epsilon^2}{2k})}
\]

\[
E[d(w_i, w_\alpha)] = \frac{nN_{\text{max}}B_{r,k}[N_{\text{max}}B_{r,k} + 1]^{n-1}}{\sum_{i=0}^{n} m_i \exp(-\frac{\epsilon^2}{2k})}
\]

**Proof:** See Appendix F.

---

5.2 Online Mechanism

For the online setting, Mechanism 2 and Algorithm 2 are modified to only generate feasible words for a Markov chain.
Mechanism 4 (Online Mechanism for Markov Chains) Fix a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and a Markov chain $(S, P, s_0)$. Given a word $w_i = s_0 \ldots s_n \in S^n$, define the online mechanism $M_{\text{on}}^{w_i}$ that chooses an output word $w_o = s'_0 \ldots s'_n \in S^n$ such that each $s'_t$ is selected from the distribution $\mathbb{P}[s'_t = \mu_{s_t}(s'_t \mid t, s^o_{t-1})]$, where $\mu_{s_t}$ is the policy synthesized by Algorithm 4, and $N(s^o_{t-1})$ is defined in Equation (2). In Algorithm 4,

$$\beta(s_t, s^o_{t-1}) = \begin{cases} 1, & \text{if } \mathbb{P}[s_t \mid s^o_{t-1}] > 0, \\ 0, & \text{otherwise}. \end{cases}$$

**Algorithm 4: Online Policy Construction for Product Modified Hamming Distance NFA**

**Input**: Probability of correct transition $\tau$, indicator function $\beta$, number of feasible states $N(s)$ for each $s \in S$

**Output**: $\mu_{s_t}$

1. for $s_t \in S$ do
2.  for $s^o_{t-1} \in S$ do
3.   for $s'_t \in S$ do
4.     if $s_t = s'_t$ and $\beta(s_t, s^o_{t-1}) = 1$ then
5.       $\mu_{s_t}(s'_t \mid s_t, s^o_{t-1}) = \tau(s_t, s^o_{t-1})$; // CT
6.     else if $s_t \neq s'_t$ and $\beta(s^o_{t-1}, s'_t) = 1$ then
7.       $\mu_{s_t}(s'_t \mid s_t, s^o_{t-1}) = (1 - \tau(s_t, s^o_{t-1})) \beta(s^o_{t-1}, s'_t) / N(s^o_{t-1})$; // ST
8.     else
9.       $\mu_{s_t}(s'_t \mid s_t, s^o_{t-1}) = 0$
10.    end
11.  end
12. end
13. end

In words, at each time $t$, only the feasible states from the most recent output symbol $s^o_{t-1}$ are assigned positive probabilities. If the current sensitive input character $s_t$ is feasible from $s^o_{t-1}$, then a correct transition is allowed with a probability $\tau(s_t, s^o_{t-1})$ and other feasible states will have identical probabilities whose sum is $1 - \tau(s_t, s^o_{t-1})$. If $s_t$ is not feasible from $s^o_{t-1}$, the online mechanism cannot select the correct transition. Then its probability is set to 0 and all feasible states are assigned equal probabilities. The time complexity of constructing the policy $\mu_{s_t}$ is $O(|S|^2)$. After its construction, one can call this policy repeatedly to generate a private state.

An example of Mechanism 4 is shown in Figure 4.

**Theorem 7 (Solution to second part of Problem 3)** Fix an adjacency parameter $k \in \mathbb{N}$ and a privacy level $\epsilon \geq 0$. For a Markov chain $(S, P, s_0)$, a sensitive input word $w_i = s_0 \ldots s_n \in S^n$ and an initial state $s_0^\epsilon$, the online mechanism $M_{\text{on}}^{w_i}$ is word-differentially private with respect to the adjacency relationship $\text{Adj}_{n,k}$ in Definition 2 if $\tau(s_t, s^o_{t-1})$ in Algorithm 4 satisfies

$$\tau(s_t, s^o_{t-1}) = \frac{1}{(N(s^o_{t-1}) - 1) \exp \left( -\frac{\epsilon}{k} \right) + 1}. \quad (4)$$

**Proof**: See Appendix G.

For a given input word $w_i$, we can bound the expectation of the distance between $w_i$ and an private output word $w_o$ by substituting $N_{\text{min}}$ (or $N_{\text{max}}$) for $N(s^o_{t-1})$ in Equation (4). The variance bound stays the same as in Theorem 6.

**Remark 5** The probability of a correct transition, $\tau$, is a function of the most recent sensitive input symbol $s_t$ and most recent output $s^o_{t-1}$ instead of being a constant as in Algorithm 2. This is because at different times the numbers of feasible states can be different. For example, in Figure 3a, if the initial output state is $s_0$, then $C(s_0) = \{s_1, s_2, s_3\}$ and $N(s_0) = 3$. But if the initial output state is $s_1$, then $C(s_1) = \{s_1, s_2\}$ and $N(s_1) = 2$.

## 6 Simulation

This section presents simulation results. Due to space constraints, only examples of Mechanism 4 are shown. The example system we use is a Markov chain model of a traffic system in Gainesville, FL that is generated by real-world traffic data. A symbolic trajectory produced by this Markov chain corresponds to a user’s route through Gainesville, and such routes are sensitive. For example, they may reveal a user’s place of home or work, their daily activities, and their acquaintances. Therefore, we implement differential privacy for this system.

To elaborate, we consider a Markov chain that is generated by the Annual Average Daily Traffic (AADT) of some of the major streets in Gainesville, Florida from 2021. The traffic data can be obtained from Florida Traffic Online (2021) [34]. Florida Traffic Online is a web-site mapping application which provides traffic count site locations and historical traffic count data. The AADT
numbers are the total volumes of traffic on a road segment for one year, divided by the number of days in that year. The Markov chain we construct contains 43 states, where each state corresponds either to a single street or a segment of a street. We compute the transition probabilities in this model using basic frequency analysis. That is, the transition probability from one state to another is equal to the number of times a driver transitions from the first state to the second divided by the total number of times a driver transitions away from the first state. The alphabet \( \Sigma \) contains one symbol for each state, and therefore \( |\Sigma| = 43 \). Figure 5 shows all streets contained in the model, and Figure 6 shows a portion of the Markov chain model along with the relevant transition probabilities.

Next we demonstrate Mechanisms 4 by generating differentially private versions of the route shown in Figure 7a.

### 6.1 Results for Different Privacy Parameters \( \epsilon \)

We illustrate the effects of privacy parameters in the range \( \epsilon \in [0.1, 10] \). Let the adjacency parameter \( k = 1 \). Figure 7 gives example private outputs for different values of \( \epsilon \). As \( \epsilon \) grows, there is a general decrease in the distance between the sensitive input route, shown in Figure 7a, and the private outputs, shown in the other subfigures. The private routes become more recognizable as \( \epsilon \) grows, and this agrees with intuition because a larger \( \epsilon \) implies weaker privacy protections and thus should provide private outputs closer to the inputs that produced them.

![Markov chain model](image)

Fig. 6. A portion of the Markov chain model used in this section. The red line (SW 2ND AVE West) is the initial state, and the arrows in different colors are the states that it is feasible to transition to from the initial state, along with the probabilities of these transitions.

### 6.2 Results for Different Initial States

We now consider different initial conditions of the Markov chain and explore how they affect the accuracy of private trajectories. Let the adjacency parameter \( k = 1 \). Figure 8 shows the average error between sensitive input words and private output words that start from three different initial states. For each initial state, 1,000 private output words were generated for each value of \( \epsilon \) to compute the average error. Figure 8 shows that for different initial conditions, Mechanism 4 will generate outputs with different average errors. Specifically, at the same level of privacy, the online mechanism starting at the state “SW 34th St” tends to make fewer errors than when starting at the state “SW Archer Rd”. This is because the state “SW Archer Rd” deviates from the correct string.

To explore this further, let \( s_t^o \) denote the random variable that is the output symbol at time \( t \), and let \( d_t \) denote the distance between the input symbol at time \( t \) and \( s_t^o \). Fix \( \epsilon = 5 \). Then for different choices of \( s_t^o \) we list the probability of a correct transition at each time \( t \in [5] \) in Table 1, i.e., we list the probability that the output symbol \( s_t^o \) is equal to the input symbol at time \( t \).

For each time \( t \), the probability of a correct transition when starting at the state “SW 34th St” is high (> 0.9). As for the initial state “SW Archer Rd”, the Markov chain dynamics force Mechanism 4 to begin with an incorrect transition (i.e., \( s_t^o \) must be different from the input symbol at time 1) because the correct transition is infeasible. We also see that beginning from “SW Archer Rd” gives Mechanism 4 a lower chance to make correct
(a) A car’s sensitive route.

(b) $\epsilon = 0.1$, error = 14

(c) $\epsilon = 1$, error = 11

(d) $\epsilon = 3$, error = 5

(e) $\epsilon = 5$, error = 1

(f) $\epsilon = 10$, error = 0

Fig. 7. Differentially private samples of a car’s route. Figure 7a shows the sensitive route itself, and each other subfigure shows a private sample of that route, with the value of $\epsilon$ and number of errors shown in its corresponding caption. All routes start from the initial state “SW 34th St”.

| $t$ | Initial State | SW 34th St | SW Archer Rd |
|-----|---------------|-------------|---------------|
| 1   |               | 0.935       | 0             |
| 2   |               | 0.993       | 0.245         |
| 3   |               | 0.967       | 0.242         |
| 4   |               | 0.954       | 0.240         |
| 5   |               | 0.948       | 0.237         |

Table 1

Probability of a correct transition at times $t \in [5]$ starting from the initial states “SW 34th St” and “SW Archer Rd”. Here $\epsilon = 5$.

This is because the mechanism assigns every feasible state the same probability of transition when the mechanism cannot make the correct transition, even though transitioning to some states would lead to fewer errors in the long run. This assignment of probabilities is done because future states cannot be known at the present time, and the errors here are due to the lack of knowledge of the future that is inherent to the online setting.

7 Conclusion

This paper presented a novel differential privacy framework for symbolic systems that generate sensitive non-numerical sequences. Differential privacy is enforced by efficient mechanisms, and these can be implemented both offline and online. These mechanisms were also extended to Markov Chains, and concentration bounds were presented to quantify the accuracy of private outputs. Future work will apply these mechanisms to Markov decision processes. By doing so, an MDP becomes a partially observable MDP (POMDP), and future work will explore privacy and performance in
reinforcement learning problems modeled with MDPs.

References

[1] F. Siddiqui, S. Zeadally, C. Alcaraz, and S. Galvao, “Smart grid privacy: Issues and solutions,” in 2012 21st International Conference on Computer Communications and Networks (ICCCN), pp. 1–5, 2012.

[2] Y. Simmhan, A. G. Kumbhare, B. Cao, and V. Prasanna, “An analysis of security and privacy issues in smart grid software architectures on clouds,” in 2011 IEEE 4th International Conference on Cloud Computing, pp. 582–589, July 2011.

[3] C. Bloom, J. Tan, J. Ramjohn, and L. Bauer, “Self-driving cars and data collection: Privacy perceptions of networked autonomous vehicles,” in Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017), pp. 357–375, 2017.

[4] M. U. Hassan, M. H. Rehmani, and J. Chen, “Differential privacy techniques for cyber physical systems: A survey,” IEEE Communications Surveys Tutorials, vol. 22, no. 1, pp. 746–789, 2020.

[5] J. Cortes, G. E. Dullerud, S. Han, J. Le Ny, S. Mitra, and G. J. Pappas, “Differential privacy in control and network systems,” in 2016 IEEE 55th Conference on Decision and Control (CDC), pp. 4252–4272, 2016.

[6] C. Dwork, “Differential privacy: A survey of results,” in International conference on theory and applications of models of computation, pp. 1–19, Springer, 2008.

[7] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” Foundations and Trends® in Theoretical Computer Science, vol. 9, no. 3–4, pp. 211–407, 2014.

[8] S. P. Kasiviswanathan and A. Smith, “On the ‘Semantics’ of differential privacy: A bayesian formulation,” Journal of Privacy and Confidentiality, vol. 6, Jun. 2014.

[9] C. Hawkins and M. Hale, “Differentially private formation control,” in 2020 59th IEEE Conference on Decision and Control (CDC), pp. 6260–6265, 2020.

[10] Y. Wang, Z. Huang, S. Mitra, and G. E. Dullerud, “Differential privacy in linear distributed control systems: Entropy minimizing mechanisms and performance tradeoffs,” IEEE Transactions on Control of Network Systems, vol. 4, no. 1, pp. 118–190, 2017.

[11] S. Han, U. Topcu, and G. J. Pappas, “Differentially private constrained optimization,” IEEE Transactions on Automatic Control, vol. 62, no. 1, pp. 50–64, 2017.

[12] M. T. Hale and M. Egerstedt, “Differentially private cloud-based multi-agent optimization with constraints,” in 2015 American Control Conference (ACC), pp. 1235–1240, 2015.

[13] E. Nozari, P. Tallafragada, and J. Cortés, “Differentially private distributed convex optimization via objective perturbation,” in 2016 American Control Conference (ACC), pp. 2061–2066, 2016.

[14] Z. Huang, S. Mitra, and N. Vaidya, “Differentially private distributed optimization,” in Proceedings of the 2015 International Conference on Distributed Computing and Networking, ICDCN ’15, (New York, NY, USA), Association for Computing Machinery, 2015.

[15] M. Hale, A. Jones, and K. Leahy, “Privacy in feedback: The differentially private LQG,” in 2018 Annual American Control Conference (ACC), pp. 3386–3391, 2018.

[16] F. Koufogiannis and G. J. Pappas, “Differential privacy for dynamical sensitive data,” in 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pp. 1118–1125, 2017.

[17] Y. Kawano and M. Cao, “Design of privacy-preserving dynamic controllers,” IEEE Transactions on Automatic Control, vol. 65, no. 9, pp. 3863–3878, 2020.

[18] J. Le Ny and G. J. Pappas, “Differentially private filtering,” IEEE Transactions on Automatic Control, vol. 59, no. 2, pp. 341–354, 2014.

[19] J. Le Ny and M. Mohammady, “Differentially private MIMO filtering for event streams,” IEEE Transactions on Automatic Control, vol. 63, no. 1, pp. 145–157, 2018.

[20] J. Stiver and P. Antsaklis, “Modeling and analysis of hybrid control systems,” in Proceedings of the 31st IEEE Conference on Decision and Control, pp. 3748–3751 vol.4, 1992.

[21] P. I. Barton, C. K. Lee, and M. Yunt, “Optimization of hybrid systems,” Computers & Chemical Engineering, vol. 30, no. 10, pp. 1576–1589, 2006. Papers form Chemical Process Control VII.

[22] C. Belta, A. Becchi, M. Egerstedt, E. Fratizzi, E. Klavins, and G. J. Pappas, “Symbolic planning and control of robot motion,” IEEE Robotics Automation Magazine, vol. 14, no. 1, pp. 61–70, 2007.

[23] P. Spiess, S. Karnouskos, D. Guiraud, D. Savio, O. Baecker, L. M. S. d. Souza, and V. Trifa, “SOA-based integration of the internet of things in enterprise services,” in 2009 IEEE International Conference on Web Services, pp. 968–973, 2009.

[24] X. Li, D. Li, J. Wan, C. Liu, and M. Imran, “Adaptive transmission optimization in SDN-based industrial internet of things with edge computing,” IEEE Internet of Things Journal, vol. 5, no. 3, pp. 1351–1360, 2018.

[25] H. Rong, X. Zhou, C. Yang, Z. Shafiq, and A. Liu, “The rich and the poor: A markov decision process approach to optimizing taxi driver revenue efficiency,” in Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp. 2329–2334, 2016.

[26] K. Akkaya, I. Guvenc, R. Aygun, N. Pala, and A. Kadri, “IoT-based occupancy monitoring techniques for...
energy-efficient smart buildings,” in 2015 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), pp. 58–63, 2015.

[27] Y. Li, L. Shi, P. Cheng, J. Chen, and D. E. Quevedo, “Jamming attacks on remote state estimation in cyber-physical systems: A game-theoretic approach,” IEEE Transactions on Automatic Control, vol. 60, no. 10, pp. 2831–2836, 2015.

[28] S. Koenig and R. Simmons, “Xavier: A robot navigation architecture based on partially observable Markov decision process models,” in Artificial Intelligence Based Mobile Robot Systems (R. B. D. Kortenkamp and R. Murphy, eds.), pp. 91 – 122, MIT Press, May 1998.

[29] B. Zhou and W. Saad, “Joint status sampling and updating for minimizing age of information in the internet of things,” IEEE Transactions on Communications, vol. 67, no. 11, pp. 7408–7428, 2019.

[30] Y. Cui, M. Kim, Y. Gu, J. jin Jung, and H. Lee, “Home appliance management system for monitoring digitized devices using cloud computing technology in ubiquitous sensor network environment,” International Journal of Distributed Sensor Networks, vol. 10, no. 2, p. 174097, 2014.

[31] D. M. Lofaro, A. Asokan, and E. M. Roderick, “Feasibility of cloud enabled humanoid robots: Development of low latency geographically adjacent real-time cloud control,” in 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), pp. 519–526, 2015.

[32] B. Yin, M. Dridi, and A. E. Moudni, “Markov decision process for traffic control at an isolated intersection,” in 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, pp. 789–794, 2013.

[33] D.-S. Zois. “Sequential decision-making in healthcare IoT: Real-time health monitoring, treatments and interventions.” in 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), pp. 24–29, 2016.

[34] “Florida traffic online (2021).” https://tdaappsprod.dot.state.fl.us/fto/.

[35] A. Jones, K. Leahy, and M. Hale, “Towards differential privacy for symbolic systems,” in 2019 American Control Conference (ACC), pp. 372–377, 2019.

[36] A. Saboori and C. N. Hadjicostis, “Current-state opacity formulations in probabilistic finite automata,” IEEE Transactions on Automatic Control, vol. 59, no. 1, pp. 120–133, 2014.

[37] B. Wu, J. Dai, and H. Lin, “Synthesis of insertion functions to enforce decentralized and joint opacity properties of discrete-event systems,” in 2018 Annual American Control Conference (ACC), pp. 3026–3031, 2018.

[38] B. Wu, Z. Liu, and H. Lin, “Parameter and insertion function co-synthesis for opacity enhancement in parametric stochastic discrete event systems,” in 2018 Annual American Control Conference (ACC), pp. 3032–3037, 2018.

[39] Y. Yin, M. Zamani, and S. Liu, “On approximate opacity of cyber-physical systems,” IEEE Transactions on Automatic Control, vol. 66, no. 4, pp. 1630–1645, 2021.

[40] Y. Savas, M. Ornik, M. Cubuktepe, and U. Topcu, “Entropy maximization for constrained Markov decision processes,” in 2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 911–918, 2018.

[41] Y. Savas, M. Ornik, M. Cubuktepe, M. O. Karabag, and U. Topcu, “Entropy maximization for Markov decision processes under temporal logic constraints,” IEEE Transactions on Automatic Control, vol. 65, no. 4, pp. 1552–1567, 2020.

[42] B. Ramasubramanian, L. Niu, A. Clark, L. Bushnell, and R. Poovendran, “Privacy-preserving resilience of cyber-physical systems to adversaries,” in 2020 59th IEEE Conference on Decision and Control (CDC), pp. 3785–3792, 2020.

[43] J. Hsu, M. Gaboardi, A. Haeberlen, S. Khamra, A. Narayan, B. C. Pierce, and A. Roth, “Differential privacy: An economic method for choosing epsilon,” 2014 IEEE 27th Computer Security Foundations Symposium, Jul 2014.

[44] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions and reversals,” Doklady Akademii Nauk SSSR, vol. 163, no. 4, pp. 845–848, 1965. Original in Russian – translation in Soviet Physics Doklady 10(8):707-710, 1966.

[45] S. Han and G. J. Pappas. “Privacy in control and dynamical systems,” Annual Review of Control, Robotics, and Autonomous Systems, vol. 1, no. 1, pp. 309–332, 2018.

[46] J. Hsu, Z. Huang, A. Roth, and Z. S. Wu, “Jointly private convex programming,” CoRR, vol. abs/1411.0998, 2014.

[47] S. Han, U. Topcu, and G. J. Pappas, “Differentially private convex optimization with piecewise affine objectives,” in 53rd IEEE Conference on Decision and Control, pp. 2160–2166, 2014.

[48] Z. Huang, S. Mitra, and G. Dullerud, “Differentially private iterative synchronous consensus,” in Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society, WPES ’12, (New York, NY, USA), p. 81–90, Association for Computing Machinery, 2012.

[49] J. Hsu, A. Roth, T. Roughgarden, and J. Ullman, “Privately solving linear programs,” in International Colloquium on Automata, Languages, and Programming, pp. 612–624, Springer, 2014.

[50] A. Todd, M. Nourian, and M. Becchi, “A memory-efficient GPU method for hamming and levenshtein distance similarity,” in 2017 IEEE 24th International Conference on High Performance Computing (HiPC), pp. 408–418, 2017.

[51] K. Schulz and S. Mihov, “Fast string correction with Levenshtein-automata,” International Journal of Document Analysis and Recognition, vol. 5, pp. 67–85, 2002.

[52] W. Hoeffding, “Probability inequalities for sums of bounded random variables,” Journal of the American Statistical Association, vol. 58, no. 301, pp. 13–30, 1963.

[53] C. McDiarmid, “On the method of bounded differences,” in Surveys in combinatorics, 1989 (Norwich, 1989), pp. 148–188, Cambridge University Press, Cambridge, 1989.

[54] R. Bhatia and C. Davis, “A better bound on the variance,” The american mathematical monthly, vol. 107, no. 4, pp. 353–357, 2000.

Appendix A  Proof of Theorem 1

We proceed by showing that Mechanism 1 implements the exponential mechanism in Definition 5 with the utility function in Definition 4. The exponential mechanism
outputs the word $w_o \in \Sigma^n$ with probability

$$p_e(w_o) = \frac{\exp \left( -\frac{ed(w_o, w)}{2k} \right)}{\sum_{w \in \Sigma^n} \exp \left( -\frac{ed(w, w)}{2k} \right)},$$

where the second equation holds since for each distance $i$, there are $\binom{n}{i} (m-1)^i$ possible outputs. For Mechanism 1, the probability of outputting the same $w_o$ is

$$p_o(w_o) = p(\ell; w_1, d(w_1, w_o)) \cdot \frac{1}{\binom{n}{\ell} (m-1) \ell d(w_1, w_o)} = \frac{\exp \left( -\frac{ed(w_o, w)}{2k} \right)}{\sum_{i=0}^{\min(n, \ell)} \binom{n}{i} (m-1)^i \exp \left( -\frac{\epsilon i}{2k} \right)}.$$

The first equation holds since each possible output word that has the same distance to the input word is equal-probable. As $p_e(w_o) = p_o(w_o)$ and the exponential mechanism is word $\epsilon$-differentially private, we conclude that the offline mechanism is word $\epsilon$-differentially private. ■

**Appendix B Proof of Theorem 2**

By expanding $E[\ell] = \sum_{i=0}^{n} p(\ell; w_i, k) \ell$ we have

$$E[\ell] = \sum_{i=0}^{n} \binom{n}{\ell} \ell (m-1)^\ell \exp \left( -\frac{\epsilon i}{2k} \right) / \sum_{i=0}^{n} \binom{n}{i} (m-1)^i \exp \left( -\frac{\epsilon i}{2k} \right),$$

which follows by plugging in $p(\ell; w_i, k)$ from Equation (1). The numerator of Equation (5) is equal to

$$\sum_{\ell=0}^{n} \binom{n}{\ell} \ell (m-1)^\ell \exp \left( -\frac{\epsilon \ell}{2k} \right) = n(m-1) \cdot \exp \left( -\frac{\epsilon \ell}{2k} \right) \sum_{i=0}^{n} \binom{n}{i} (m-1)^i \exp \left( -\frac{\epsilon i}{2k} \right),$$

which follows by factoring out $n(m-1) \exp \left( -\frac{\epsilon \ell}{2k} \right)$ and using the binomial theorem on the resulting sum. The binomial theorem can also be used for the denominator of Equation (5). That result and Equation (6) give the expectation.

For variance, expanding $E[\ell^2] = \sum_{i=0}^{n} p(\ell; w_i, k) \ell^2$ gives

$$E[\ell^2] = \sum_{i=0}^{n} \binom{n}{\ell} \ell (m-1)^\ell \exp \left( -\frac{\epsilon \ell}{2k} \right) / \sum_{i=0}^{\min(n, \ell)} \binom{n}{i} (m-1)^i \exp \left( -\frac{\epsilon i}{2k} \right).$$

The denominator of Equation (7) can be simplified using the binomial theorem. The numerator can be simplified by factoring out $n(m-1) \exp \left( -\frac{\epsilon \ell}{2k} \right)$, which then gives

$$\sum_{\ell=0}^{n} \binom{n}{\ell} \ell^2 (m-1)^\ell \exp \left( -\frac{\epsilon \ell}{2k} \right) = n(m-1) \exp \left( -\frac{\epsilon \ell}{2k} \right) \cdot [b_1 + b_2],$$

where $b_1 = \sum_{i=0}^{n-1} \binom{n-1}{i} (m-1)^i \exp \left( -\frac{\epsilon i}{2k} \right)$ and $b_2 = \sum_{i=0}^{n-1} \binom{n-1}{i} (m-1)^i \exp \left( -\frac{\epsilon i}{2k} \right)$. Then $b_1$ can be simplified the same way as Equation (6), and $b_2$ can be simplified using the binomial theorem. Then $E[\ell^2]$ is

$$n(m-1) \exp \left( -\frac{\epsilon \ell}{2k} \right) \sum_{i=0}^{n} \binom{n}{i} (m-1)^i \exp \left( -\frac{\epsilon i}{2k} \right) + 1,$$

and we use $Var[\ell] = E[\ell^2] - E[\ell]^2$. Concentration bounds follow from Chernoff-Hoeffding bounds [52]. ■

**Appendix C Proof of Theorem 3**

The given value of $\tau$ satisfies $\tau > \frac{1}{m-1}$, which follows from $\tau - \frac{1}{m-1} = \frac{1 - \exp(-\epsilon/k)}{\epsilon} > 0$. Then for all $(w_i, w_i') \in \text{Adj}_{n,k}$ and any output word $w_o$, we have $p(w_o; w_i) = \frac{\exp(-\epsilon/k)}{\tau^n-d(w_i, w_o)}$, because each output character is chosen independently. A similar statement holds for $p(w_o; w_i')$. Let $d = d(w_i, w_o) - d(w_i', w_o)$. From $(w_i, w_i') \in \text{Adj}_{n,k}$, it follows that $-k \leq d \leq k$. Then

$$\frac{p(w_o; w_i) - p(w_o; w_i')}{\exp(-\epsilon/k)} = \frac{1 - \tau^d}{\tau^n - d(w_i, w_o)} = \frac{1 - \tau^d}{(m-1)\tau^n} = \exp(\epsilon),$$

where the first inequality holds because $\frac{1 - \tau^d}{(m-1)\tau^n} < 1$, and $d \geq -k$. The final equality holds by plugging in $\tau$. Showing $\mathbb{E}[d(w_i, w_o) \mid \ell] = \binom{n}{\ell} (m-1)^\ell \exp \left( -\frac{\epsilon \ell}{2k} \right)$ uses the same technique. ■

**Appendix D Proof of Theorem 4**

In Mechanism 2, $d(w_i, w_o)$ has the distribution $\mathbb{P}[d(w_i, w_o) = \ell] = \binom{n}{\ell} (m-1)^\ell \left( \frac{1 - \tau^d}{(m-1)\tau^n} \right)^\ell$, which is Binomial. Then the result follows from $E[d(w_i, w_o)] = n(1 - \tau), Var[d(w_i, w_o)] = n\tau(1 - \tau)$, and expanding $\tau$. Concentration bounds follow from Chernoff bounds [53]. ■
Appendix E  Proof of Theorem 5

Like the proof of Theorem 1 in Appendix A, this proof proceeds by showing that Mechanism 3 implements the exponential mechanism in Definition 5. For the input word \( w_1 \), we will verify the equality of the probability with which the exponential mechanism outputs \( w_o \), which is \( p_o(w_o) \), and the probability with which that same word is output by Mechanism 3, which is \( p_o(w_o) \).

Let \( F(s_0) \subseteq S^n \) be the set of all feasible words with initial state \( s_0 \). Suppose that \( d(w_i, w_o) = \ell \). Then

\[
p_{\epsilon}(w_o) = \frac{\exp \left( -\frac{\epsilon \ell}{2k} \right)}{\sum_{w' \in F(s_0)} \exp \left( \frac{-\epsilon d(w, w')}{2k} \right)} = \frac{\exp \left( -\frac{\epsilon \ell}{2k} \right)}{\sum_{i=0}^{n} m_i \exp \left( -\frac{\epsilon i}{2k} \right)} = p_o(w_o).
\]

where \( m_i \) is the number of words in \( F(s_0) \) that are distance \( i \) from \( w_i \).

Mechanism 3 outputs the same \( w_o \) with probability

\[
p_o(w_o) = p(\ell; w_i, k) \cdot \frac{1}{m_{\ell}} = \frac{\exp \left( -\frac{\epsilon \ell}{2k} \right)}{\sum_{i=0}^{n} m_i \exp \left( -\frac{\epsilon i}{2k} \right)} = p_o(w_o).
\]

The first equation holds since each possible output word that has the same distance to the input word is equal-probable. The second equation expands \( p(\ell; w_i, k) \) and simplifies. Since the exponential mechanism is word \( \epsilon \)-differentially private, Mechanism 3 is, too.

Appendix F  Proof of Theorem 6

Given \( w_i \in S^n \), we have \( E[d(w_i, w_o)] = \frac{\sum_{t=0}^{n} \ell(t) m_t \exp \left( -\frac{\epsilon t}{2k} \right)}{\sum_{i=0}^{n} m_i \exp \left( -\frac{\epsilon i}{2k} \right)} \).

For every \( \ell, (\ell)_{i} \), \( (N_{\min} - 1)^{\ell} \leq m_t \leq (\ell)_{i} N_{\max}^{\ell} \). Then

\[
\sum_{t=0}^{n} \ell(t) (N_{\min} - 1)^{\ell} \exp \left( -\frac{\epsilon t}{2k} \right) \leq \sum_{t=0}^{n} \ell(t) N_{\max}^{\ell} \exp \left( -\frac{\epsilon t}{2k} \right) \leq E[d(w_i, w_o)] \leq \frac{\sum_{t=0}^{n} \ell(t) N_{\max}^{\ell} \exp \left( -\frac{\epsilon t}{2k} \right)}{\sum_{i=0}^{n} m_i \exp \left( -\frac{\epsilon i}{2k} \right)}. \tag{8}
\]

Equation (8) can be further simplified using the same procedure to reach Equation (6). The variance bound follows from Popoviciu’s inequality [54].

Appendix G  Proof of Theorem 7

Let \( w_1 = s_{11}s_{22} \ldots s_{1n} \in S^n \) be a sensitive input word and let \( w_2 = s_{21}s_{22} \ldots s_{2n} \in S^n \) be another such that \( d(w_1, w_2) = k \). Consider \( M_{w,s}^{(m)}(w_1) = M_{w,s}^{(m)}(w_2) = w_o \) and \( w_o = s_{11}^{(o)} s_{12}^{(o)} \ldots s_{1n}^{(o)} \in S^n \). Then

\[
\begin{align*}
\mathbb{P}[M_{w,s}^{(m)}(w_1) = w_o] &= \prod_{i=1}^{t} \mu_{\epsilon,s}(s_{1i}^{(o)} | s_{1i}^{(o)} - 1) \\
\mathbb{P}[M_{w,s}^{(m)}(w_2) = w_o] &= \prod_{i=1}^{t} \mu_{\epsilon,s}(s_{2i}^{(o)} | s_{2i}^{(o)} - 1) \\
&= \prod_{i \in B} \mu_{\epsilon,s}(s_{2i}^{(o)} | s_{2i}^{(o)} - 1), \tag{9}
\end{align*}
\]

where \( B = \{ j \in [n] | s_{1j} \neq s_{2j} \} \) and \(|B| \leq k \). Note that in Algorithm 4, we have

\[
\mu_{\epsilon,s}(s_{1i}^{(o)} | s_{1i}^{(o)} - 1) = \begin{cases} 
1 - \tau(s_{1i}^{(o)}, s_{1i}^{(o)} - 1) & \text{if } \beta(s_{1i}^{(o)}, s_{1i}^{(o)} - 1) = 1 \\
\frac{1}{N(s_{1i}^{(o)} - 1)} & \text{otherwise}
\end{cases}
\]

We next show that Equation (4) gives \( \tau(s_{1i}^{(o)}, s_{1i}^{(o)} - 1) \geq \frac{1}{N(s_{1i}^{(o)} - 1)} \). The first inequality follows from

\[
\tau(s_{1i}^{(o)}, s_{1i}^{(o)} - 1) = \frac{1}{N(s_{1i}^{(o)} - 1)} \exp \left( -\frac{\epsilon}{k} \right) + \frac{1}{N(s_{1i}^{(o)} - 1)} \exp \left( -\frac{\epsilon}{k} \right).
\]

The second equation holds by regrouping the two terms and the inequality holds since \( \exp \left( -\frac{\epsilon}{k} \right) \leq 1 \). We next prove that \( \frac{1}{N(s_{1i}^{(o)} - 1)} \geq \frac{1}{N(s_{1i}^{(o)} - 1)} \). Because we have

\[
\frac{1}{N(s_{1i}^{(o)} - 1)} - \frac{1}{N(s_{1i}^{(o)} - 1)} = \frac{N(s_{1i}^{(o)} - 1) - 1}{N(s_{1i}^{(o)} - 1)} \leq 0.
\]

The second equation holds by factoring and the third holds by plugging in \( \tau(s_{1i}^{(o)}, s_{1i}^{(o)} - 1) \). The final inequality holds since \( \exp \left( -\frac{\epsilon}{k} \right) \leq 1 \). Now we bound Equation (9) with

\[
\prod_{i \in B} \mu_{\epsilon,s}(s_{1i}^{(o)} | s_{2i}^{(o)} - 1) \leq \prod_{i \in B} \frac{\tau(s_{1i}^{(o)}, s_{1i}^{(o)} - 1)}{N(s_{1i}^{(o)} - 1)} \exp \left( -\frac{\epsilon}{k} \right) = \prod_{i \in B} \exp \left( -\frac{\epsilon}{k} \right). \tag{10}
\]

Equation (10) holds by expanding \( \tau \) at each state using \(|B| \leq k \). The bound \( \prod_{i \in B} \mu_{\epsilon,s}(s_{1i}^{(o)} | s_{2i}^{(o)} - 1) \geq \exp(-\epsilon) \) follows using the same technique.