Differential Privacy to the Rescue? On Obfuscating Tolls in Privacy-Preserving ETC Systems

Valerie Fetzer
Karlsruhe Institute of Technology and KASTEL SRL
Germany
valerie.fetzer@kit.edu

Amirhossein Adavoudi Jolfaei
University of Luxembourg
Luxembourg
amirhossein.adavoudi@uni.lu

Andy Rupp
University of Luxembourg and KASTEL SRL
Luxembourg
andy.rupp@uni.lu

Stefan Schiffner
Berufliche Hochschule Hamburg (BHH)
Germany
Stefan@inf-bhh.de

ABSTRACT

Electronic toll collection (ETC) systems are becoming increasingly popular, but are inherently privacy-sensitive as they deal with users' location data. To this end, quite some research effort has been put into the design of privacy-preserving ETC (PPETC) systems. In this paper, we study the actual privacy properties of PPETC schemes, which hide the individual toll fees from the toll service provider and provide it only with a total monthly fee. Since previous work has shown that PPETC schemes may not suffice to protect the privacy of users in real scenarios, we analyze the effectiveness of using an additional protection mechanism: applying a differential privacy mechanism that hides the actual monthly toll fee by adding a small amount of noise. While this seems like a straightforward solution, it is not that simple: Since adding noise to monthly fees can increase monetary costs for users, the added noise should be kept small. But since adding more noise intuitively means more privacy when applying differential privacy, one must carefully choose the amount of added noise in order to strike a balance between privacy gain and additional cost.

Our goal is to examine two popular differential privacy mechanisms, namely d-privacy and the exponential mechanism, in order to evaluate their effectiveness in protecting a user's toll station visits and to determine the associated privacy costs. To investigate how well they hide the visited toll stations, we design for each protection mechanism an attack mechanism that attempts to recover those from a obscured monthly toll fee, and evaluate its effectiveness on two real-world scenarios.

KEYWORDS

Electronic toll collection, differential privacy, cost, metric privacy, exponential mechanism

1 INTRODUCTION

Electronic toll collection (ETC) is a technology that is primarily used to finance road infrastructure, but can also be used for advanced functions, such as congestion management and pollution reduction through dynamic pricing. ETC systems are implemented by tolling service providers (TSPs), which are authorized to collect tolls and manage the tolling system and are often private companies. In this paper, we focus on *post-payment* ETC systems with monthly billing periods, as these systems are more convenient for users than pre-payment systems. In post-payment ETC systems, the TSP needs to store certain sensitive information in order to be able to charge users, i.e., users' names, billing addresses, payment information, and monthly toll charges. In practice, however, fine-grained billing information, such as the exact times (and locations) of toll station visits, is also stored, which inherently allows the TSP to track the movements of each user. This issue has long been known in the research community [28]. To address this privacy issue, several *privacy-preserving ETC (PPETC)* schemes have been developed [4, 18, 21, 23, 29] that minimize information leakage to the TSP while still allowing users to be charged.

However, research indicates that simply implementing PPETC schemes may not sufficiently protect privacy [2, 8, 10]. This is because there is still *some* information leakage to the TSP, such as the monthly toll fee, which is needed for billing. In [2], it is shown that the monthly toll fee, combined with publicly available background data such as road maps and usage statistics, is, in some cases, sufficient to violate user privacy. More specifically, an attack is constructed on the real ETC system deployed in Brisbane that reveals the toll stations visited by users with a monthly toll fee of $\leq$10 dollars with a success rate of 94%. This attack stems from the observation that, given the monthly billing fee, reconstructing the visited toll station is equivalent to the well-known *subset sum problem (SSP)*. While the SSP is NP-complete from a complexity-theoretic point of view, it may still be efficiently solvable for "small" instances, such as the Brisbane ETC system.

Using Differential Privacy to Restore Privacy. While the concept of using *differential privacy (DP)* to obscure monthly billing fees for enhanced privacy has already been proposed [10], it overlooks the critical consideration of the *cost of privacy*. This is particularly important in ETC systems, where cost is a major concern for both TSPs and their customers [22]. We address this gap by investigating whether user privacy can be protected while ensuring that the associated cost remains relatively small. To accomplish this, we consider two DP-based protection mechanisms for ETC systems and evaluate them in two real-world ETC infrastructures. We assume that users themselves apply a DP mechanism before submitting their

(), 1–24

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.

monthly fee to the TSP such that the TSP learns only the obscured fee and not the original one.[1]

Trade-Off between Privacy & Utility. When employing the DP framework, one must always carefully balance the *privacy* of the users against the *utility* (= accuracy) of the noisy data. It is easy to see that the higher the noise, the higher the privacy of the users, but the lower the utility/accuracy of the data, and vice versa. In the context of ETC, the addition of noise also corresponds to a higher billing fee that a user has to pay.[2] How much users are willing to pay for the privacy of their data is an independent research question that does not seem to be fully answered yet. While the monetary value of privacy has been empirically evaluated for "some" contexts a few years ago (e.g., online privacy [20], location data privacy [3, 9, 38], or removal from marketers' call lists [36]), [1] suggests that this question is not easy for many people to answer and is highly context-dependent. In this work, we, therefore, do not make assumptions about the costs users are willing to bear for their data privacy, as this is a separate line of research. Instead, we focus on developing a mechanism to hide the exact monthly toll fees from the TSP, while evaluating how much noise needs to be added to achieve ε-DP for a given ε.

Our Contribution. We examine two differential privacy mechanisms, namely d-privacy [7] and the exponential mechanism [27], each of which provides a different granularity of privacy. We evaluate their effectiveness in protecting a user's toll station visits from an adversary by simulating attacks on the obscured monthly toll fee and evaluating their success chance. We also examine the cost the users have to pay for hiding their toll fee.

Our results show that for $\varepsilon \leq 1$, both mechanisms achieve ε-DP and successfully prevent an adversary that uses a maximum likelihood approach [31, 33] from determining the visited toll stations. However, the monthly cost coming with this may be multiple dollars. When considering a less stringent privacy notion, i.e., ε-DP for $\varepsilon = 5$, the adversary can determine the visited toll stations with a higher, but still small, chance, with significantly lower costs. Whether or not this obfuscation approach is suitable for real-world ETC schemes depends on the price the users are willing to pay for their privacy.

2 BACKGROUND

We introduce some terms and concepts used in our ETC scenario and describe our DP frameworks. An overview of the used variables can be found in Appendix A.

2.1 ETC Background

We introduce several notions that are used in our ETC scenario. Note that we adopt some notions from [2].

Billing Period: We assume users pay their tolls once per billing period, e.g., once per month.

Toll Stations: We use $S = \{s_1, s_2, \ldots, s_l\}$ as set of toll stations.

Pricing Model: We define the pricing model of an ETC scheme as a set of toll prices $P = \{p_1, p_2, \ldots, p_l\}$, where each price p_j is fixed and assigned to toll station s_j.

Trace: A trace records the toll stations visited by a user during a billing period, including the frequency [2]. A trace is denoted as $trace = \{(s_1, f_1), (s_2, f_2), \ldots, (s_l, f_l)\}$, where f_i is the frequency associated with the toll station s_i.

Wallet: A wallet represents the state of a user at the end of a billing period. It consists of the trace $trace$ and the wallet balance[3] w, i.e., the sum of all prices of the visited toll stations. Given w, the following equation holds:

$$w = p_1 \cdot f_1 + p_2 \cdot f_2 + \cdots + p_l \cdot f_l, \quad f_j \in \mathbb{N}_0$$

Plausible Wallets: The set of plausible wallets is the set of all wallets that can be possibly achieved, given a pricing model P. To determine the plausible wallets falling within the range of $[w_l, w_u]$, we formulate the following inequality and find all solutions within this range.

$$w_l < p_1 \cdot f_1 + p_2 \cdot f_2 + \cdots + p_l \cdot f_l < w_u \tag{1}$$

The set of all solutions derived from Inequality 1 is denoted as W_p, where each element $(id, w, trace)$ consists of a wallet id id, a wallet balance w, and a trace $trace$. Note that we add ids to wallets here to be able to differentiate between wallets that have the same balance, but different traces. Note that the set of plausible wallets could then be further refined by using information about the road network and connectivity between toll stations. Solutions from Inequality 1 can be discarded if they are not possible given the road network.

Plausible Trace: A plausible trace is a trace that can be possibly achieved by a user and is linked with a plausible wallet. The set of plausible traces is defined by $T_p \coloneqq \{trace \mid (\cdot, \cdot, trace) \in W_p\}$.

Cost of Privacy: We define the *cost* as the difference between the balance of the original wallet and the balance of the obfuscated wallet.

Subset Sum Problem (SSP): The SSP is an NP-complete problem [25], where we consider a set $A = \{a_j : 1 \leq j \leq k, a_j \in \mathbb{N}_0\}$ and a value $M \in \mathbb{N}_0$, i.e., a non-negative integer. The aim is to find x_is such that $a_1 \cdot x_1 + a_2 \cdot x_2 + \cdots + a_k \cdot x_k = M, x_j \in \mathbb{N}_0$.

2.2 Differential Privacy

Differential privacy (DP) was introduced as a standard for protecting personal records within datasets. The intuition behind DP is that the presence or absence of a record in a dataset should not significantly modify the statistics extracted from the dataset [11], and the information leakage from the statistics should be negligible. By doing so, the privacy of each individual record will be preserved.

We now review some terms used in the context of DP.

2.2.1 *Differential Privacy Fundamentals.*

Definition 2.1 (Distance)**.** The *distance* $Dist(D_1, D_2)$ between two datasets $D_1 \in \mathcal{D}$ and $D_2 \in \mathcal{D}$ equals the number of records where D_1 and D_2 differ.

Definition 2.2 (ε-differential privacy)**.** A randomized mechanism K gives *ε-differential privacy (ε-DP)* if for all datasets $D_1, D_2 \in \mathcal{D}$

[1] We assume that the user also appends a zero-knowledge proof that they applied the DP mechanism correctly. Details on this can be found in Remark 2.

[2] Since we assume that the noise can also be negative, there is a possibility that the TSP will lose some of its revenue. Therefore, it is also in the TSP's interest to keep the noise small.

[3] Note that we will frequently abbreviate the wallet balance with just wallet.

differing on at most one record, i.e., with $Dist(D_1, D_2) \leq 1$, and all $S \subseteq Range(K)$ it holds that

$$\Pr[K(D) \in S \mid D = D_1] \leq e^{\varepsilon} \cdot \Pr[K(D) \in S \mid D = D_2].$$

Intuitively, mechanism K provides ε–differential privacy if adding or removing a single record in a dataset only affects the probability of any output by a small multiplicative factor [32]. The privacy parameter ε quantifies the level of privacy loss. A smaller value of ε indicates less privacy loss [13].

In our case, the dataset will be the set of users' wallet balances.

Definition 2.3 (Global sensitivity)**.** The *global sensitivity* of a query function $f : D \rightarrow \mathbb{R}^q$ is the maximum distance between the values of the function for $\forall$ D_1 and D_2 differing in at most one record, i.e., $\Delta_f \coloneqq max||f(D_1) - f(D_2)||_1$, where $Dist(D_1, D_2) \leq 1$, and $D_1, D_2 \in \mathcal{D}$. It's important to note that sensitivity is a characteristic of the function itself and is not influenced by the database [12].

Definition 2.4 (Laplace mechanism)**.** Dwork et al. [14] demonstrated that ε-DP can be obtained by adding independent and identically distributed noise to the output of query f. The noise x is specifically sampled from the Laplace distribution ($Lap()$), whose probability density function (pdf) is denoted as $K(x) \coloneqq \frac{1}{2\lambda} e^{\frac{-|x-\mu|}{\lambda}}$, where μ is a mean and λ is a scale factor. Dwork et al. prove that adding noise from $Lap(\frac{\Delta_f}{\varepsilon})$ to an output of query f with global sensitivity Δ_f gives ε–differntial privacy.

2.2.2 *Metric Differential Privacy (d-Privacy).*

Definition 2.5 (Metric differential privacy (d_x-privacy))**.** Standard differential privacy makes it challenging to fully protect a value without adding an excessive amount of noise. However, by adopting a more relaxed metric, we can ensure a meaningful privacy guarantee by maintaining the accuracy of the values. Metric privacy helps protect the accuracy of a value [7], such as a wallet balance in our case. This is particularly useful, as disclosing the exact value of a wallet balance could reveal information about a user's movements, e.g., a trace [2]. Differential privacy can be extended to apply to any set of secrets X, provided it is associated with a metric d_X [7]. In our case, we define the metric d_x as the Euclidean distance between two wallet balances. Let $\mathcal{F}_{\mathcal{Z}}$ be a σ-algebra over $\mathcal{Z}$ and let $\mathcal{P}(\mathcal{Z})$ be the set of probability measures over $\mathcal{Z}$. A metric on a set X is a function $d_X : X^2 \rightarrow [0, \infty)$ such that $d_X(x, y) = 0$ if and only if $x = y$, $d_X(x, y) = d_X(y, x)$, and $d_X(x, z) \leq d_X(x, y) + d_X(y, z)$ for all $x, y, z \in X$. A mechanism $K : X \rightarrow \mathcal{P}(\mathcal{Z})$ satisfies ε-d_X-privacy, iff $\forall x, x' \in X$ and $\varepsilon \geq 0$:

$$K(x)(Z) \leq e^{\varepsilon \cdot d_X(x,x')} K(x')(Z) \quad \forall Z \in \mathcal{F}_{\mathcal{Z}}$$

Intuitively, the definition implies that secrets that are hardly indistinguishable with respect to d_X should yield outcomes with nearly the same likelihood [7]. In general, mechanisms developed for differential privacy can be adapted for metric differential privacy by using an appropriate metric for the domain [17]. In our case, we use the Laplace mechanism (see Definition 2.4) to provide d_x-privacy. We prove in Appendix B.1 how the Laplace mechanism ensures ε-d_x-privacy.

***Parameters** RE, z, pr.* We also provide a reasonable *relative error bound RE* while applying metric differential privacy, meaning that the extra cost should be tolerable w.r.t users with minimum wallet balances. We measure the relative error as $RE \coloneqq \frac{|w_o - w|}{w} = \frac{|N|}{w}$, where N is the amount of noise added to w for obfuscation [37, 39].

We relate RE to privacy (ε), given parameters such as sensitivity (Δ) and Laplace's scale (λ). To this end, we calculate the probability pr of a random Laplace noise exceeding the *maximum noise* z. The following integral computes the cumulative distribution function for the Laplace distribution over the interval $[-z, z]$:

$$\Pr(-z \leq x \leq z) \coloneqq \int_{-z}^{z} \frac{1}{2\lambda} e^{\frac{-|x|}{\lambda}} \, dx = 1 - e^{\frac{-z}{\lambda}}$$

From this, we derive $\Pr(|x| \geq z) \coloneqq e^{\frac{-z}{\lambda}}$. Let $pr \coloneqq \Pr(|x| \geq z)$, meaning that with probability pr, the random noise has an absolute value of at least z or, in other words, with probability $1 - pr$, the noise has a value of at maximum z. We thus call pr the *out-of-bounds probability*, since it denotes the probability that the generated noise is outside the bound defined by z.

The maximum noise z for a given pr is then obtained as $z \coloneqq -\lambda \cdot ln(pr)$. Given $\lambda \coloneqq \frac{\Delta}{\varepsilon}$ and $z \coloneqq -\lambda \cdot ln(pr)$, the following holds:

$$z \coloneqq -\frac{\Delta}{\varepsilon} \cdot ln(pr) \tag{2}$$

To ensure that the relative error remains below the specified threshold denoted as RE for every wallet w in the set W, the condition $RE \coloneqq \frac{z}{w_{min}}$ must be satisfied, where w_{min} is the smallest w in the set W. It is evident that the relative error will be less than and equal RE for all $w \geq w_{min}$. Given $RE \coloneqq \frac{z}{w_{min}}$, and Eq. (2), we can derive the following:

$$RE \coloneqq \frac{z}{w_{min}} = \frac{-\Delta \cdot ln(pr)}{\varepsilon \cdot w_{min}} \tag{3}$$

Figure 3 in Appendix B.2 shows how different parameters are connected in Eq. (3). One interpretation of Eq. (3) is that to guarantee the relative error RE, the ε of the Laplace mechanism should be derived as

$$\varepsilon \coloneqq \frac{-\Delta \cdot \ln(pr)}{RE \cdot w_{min}}. \tag{4}$$

***Post-processing**.* Let K be an ε–DP mechanism, and g be an arbitrary mapping from the set of possible outputs to an arbitrary set. Then, $g \circ K$ is ε-differentially private as well [16, 40]. In our application, when dealing with obfuscated wallet balances, we clamp them to the interval $[0, w_{max}]$, where w_{max} is the maximum wallet balance, and round to cents.

2.2.3 *The Exponential Mechanism.*

Definition 2.6 (Exponential Mechanism)**.** The exponential mechanism provides more fine-grained privacy. Instead of merely obfuscating wallet balances through the d-privacy mechanism, it allows us to adjust the level of privacy by obfuscating traces, including both the toll stations visited and their frequencies, using a scoring function. With a privacy parameter ε, an outcome set R, and a *scoring function* $u : D \times R \rightarrow \mathbb{R}$ which maps $(x \in D, r \in R)$ pairs to a real-valued score, the exponential mechanism $M_E(x, u, R)$ samples a single element r from R according to the following probability

distribution [27]:

$$\Pr[r] = \frac{\exp\left(\frac{\varepsilon \cdot u(x,r)}{2 \cdot \Delta_u}\right)}{\sum_{i \in R} \exp\left(\frac{\varepsilon \cdot u(x,i)}{2 \cdot \Delta_u}\right)} \tag{5}$$

The goal is to select a candidate item $r \in R$ that approximately maximizes $u(x, r)$ while ensuring ε-differential privacy. The sensitivity of the scoring function u is defined as

$$\Delta_u = \max_{r \in R} \max_{x, x'} |u(x, r) - u(x', r)| \tag{6}$$

where x and x' are neighboring datasets [27]. In our case, $D = R$, which equals the set of plausible traces T_p. Note that we assume all traces to be "neighbors", thus aiming to hide the complete trace from the adversary. We aim to map an original $trace \in T_p$ to an obfuscated $trace \in T_p$ using the u function. The cost is calculated as the absolute difference between the obfuscated wallet and the original wallet associated with the obfuscated and original traces.

3 WALLET OBFUSCATION MECHANISMS

In post-payment ETC systems, users often settle their debt once per billing period, i.e., they clear their wallet balance, which corresponds to the sum of all prices of all toll stations visited during that billing period. It is shown in [2, 8] that revealing the exact wallet balance can violate user privacy, since a user's trace may be recovered with significant probability with appropriate additional information. To address this issue, we present two different DP-based *wallet obfuscation mechanisms*.

3.1 Obfuscation based on d-Privacy

We first present a wallet obfuscation mechanism based on d-privacy (cp. Definition 2.5), which obscures the final wallet balance. The wallet obfuscation algorithm is shown in Algorithm 6 in Appendix C.1 and gets the user's actual wallet balance w, the minimum possible wallet balance w_{min}, the maximum possible wallet balance w_{max}, the relative error threshold RE, and the out-of-bounds probability pr as input. Note that w_{min} and w_{max} are defined by the pricing scheme and thus fixed for a given ETC scheme and that the privacy level ε is fixed through RE, pr, and w_{min} (cp. Eq. (4)).

The algorithm first sets the scale λ of the Laplace mechanism using Eq. (4) and the relation $\lambda = \frac{\Delta}{\varepsilon}$ as follows:

$$\lambda = \frac{\Delta}{\varepsilon} = \frac{\Delta}{-(\Delta \cdot ln(pr))/(RE \cdot w_{min})} = -(RE \cdot w_{min})/ln(pr)$$

Then it generates noise N using the Laplace mechanism with scale λ and adds N to the wallet balance w to get the obfuscated wallet balance w_o. Afterward, post-processing is performed: The obfuscated wallet balance w_o is clamped to the interval $[0, w_{max}]$ and rounded to two decimal places (since wallet balances are expressed in dollars and cents). Finally, the algorithm returns w_o.

We want to highlight that the generated noise N falls into interval $(-z, z)$, with probability $1 - pr$, ensuring the additional costs are capped at RE percent of w_{min} with overwhelming $(1 - pr)$ probability (cp. Eq. (3)).

Remark 1. Algorithm 6 is intended to be run by the *user* after each billing period, so that only w_o and not w is sent to the TSP for billing purposes. As such, the user needs appropriate information to sample the noise according to the TSP's intended distribution. Instead of providing the user with the obfuscation parameters (w_{min}, RE, pr), it is sufficient to provide the user with λ, since Algorithm 6 uses the obfuscation parameters only to compute λ.

Remark 2. We assume that Algorithm 6 is used in combination with *privacy-preserving* ETC schemes. These should already use cryptographic methods to ensure that a user's final wallet balance w correctly equals the sum of the individual toll fees, without the TSP learning the individual toll fees. Since the TSP now learns w_o instead of w, the TSP must again ensure that w_o represents the correct wallet balance. To achieve this, the user can, for example, send w_o together with a zero-knowledge proof (proving that w_o was computed from w using Algorithm 6 for some w[4]) to the TSP. Note that if users are not trusted to sample $N \leftarrow Lap(\lambda)$ honestly, the user and the TSP could engage in a joint coin toss to sample the noise N together [24].

Impact of Parameters on Generated Noise. In Appendix C.2 we discuss how the out-of-bounds probability pr and minimum wallet balance w_{min} impact the generated noise, which defines the extra amount of cost users have to bear for privacy. In a nutshell, we show that TSPs either need to fix a privacy goal (ε) and then see how much noise they need to achieve that or they need to fix a noise bound z and see how much privacy they achieve with that.

3.2 Obfuscation based on the Exponential Mechanism

Next, we construct an obfuscation mechanism based on the exponential mechanism M_E (cp. Definition 2.6), which operates on traces directly. The pseudocode of the mechanism is shown in Algorithm 1. The core idea is that, given a fixed trace $trace \in T_p$, M_E selects a trace from T_p that maximizes the score while guaranteeing privacy. Our scoring function $u : T_p \times T_p \to \mathbb{R}$ uses both Euclidean and similarity distances to calculate the score of two traces:

(1) The *Euclidean distance* d_{eucl} measures the difference between the two wallet balances associated with the traces
(2) The *similarity distance* d_{sim} measures how similar the traces are in terms of toll station visits

For a high score, the Euclidean distance should be small (low cost), while the similarity distance should be high (very different traces). To compute the final score, we assign weights α_{eucl} and α_{sim} (with $\alpha_{eucl} + \alpha_{sim} = 1$) to both distances, allowing a TSP to adjust the impact of each distance on the score. Then we compute the score as $score \coloneqq (d'_{sim} \cdot \alpha_{sim} - d'_{eucl} \cdot \alpha_{eucl})$, where d'_{sim} (resp. d'_{eucl}) is d_{sim} (resp. d_{eucl}) scaled to the range $[0, 1]$. Given $score$, we compute the probability that $trace_j \in T_p$ is selected by M_E as $prob \coloneqq e^{\frac{\varepsilon \cdot score}{2 \cdot \Delta}}$, where Δ is the sensitivity of the scoring function (in our case, $\Delta \coloneqq 1$). After doing this for all $trace_j \in T_p$, M_E samples an obfuscated trace according to the normalized probabilities $prob$. Note: The TSP can enhance the scoring function by incorporating additional parameters, enabling a more precise and customizable level of privacy granularity. Additional details on M_E are given in Appendix D.

[4]For w, the correctness is already ensured by the PPETC scheme (which could also use a zero-knowledge proof to achieve this).

Algorithm 1 Obfuscation Algorithm based on the Exponential Mechanism

Input: $trace \in T_p, T_p, \varepsilon, \alpha_{eucl} \in [0,1], \alpha_{sim} \in [0,1]$
Output: obf_trace

function EXPONENTIAL_OBFUSCATION($trace, T_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}$)
  $(max\_eucl, max\_sim) \leftarrow$ COMPUTE_MAX_DIST(T_p)
  Declare $arr\_score[|T_p|]$
  for all $trace_j \in T_p$ **do**
    $d_{eucl} \leftarrow$ COMPUTE_EUCLIDEAN($trace, trace_j$)
    $d_{sim} \leftarrow$ COMPUTE_SIMILARITY($trace, trace_j$)
    $d'_{eucl} \leftarrow d_{eucl}/max\_eucl$
    $d'_{sim} \leftarrow d_{sim}/max\_sim$
    $arr\_score[j] \leftarrow (d'_{sim} \cdot \alpha_{sim} - d'_{eucl} \cdot \alpha_{eucl})$
  end for
  $arr\_prob \leftarrow$ COMPUTE_PROB($arr\_score, \varepsilon, \Delta \coloneqq 1$)
  $arr\_norm\_prob \leftarrow$ NORMALIZE(arr_prob)
  $obf\_trace \leftarrow$ SELECT_RAND(arr_norm_prob, T_p)
  return obf_trace
end function

Algorithm 2 Precomputation for Attack on d-Privacy

Input: W_p, RE
Output: $list\_ranges$

function PRECOMPUTATION_METRIC_ATTACK(W_p, RE)
  $z \leftarrow RE \cdot w_{min}$
  for all $w \in W_p$ **do**
    $l \leftarrow w - z$
    $u \leftarrow w + z$
    if $l < 0$ **then**
      $l \leftarrow 0$
    end if
    if $u > w_{max}$ **then**
      $u \leftarrow w_{max}$
    end if
    $R \leftarrow (l, u)$
    $list\_ranges \leftarrow (R, w)$
  end for
  return $list\_ranges$
end function

4 DEOBFUSCATION ATTACKS

To measure the effectiveness of the DP-based wallet obfuscation mechanisms, we design attacks against them in this section and evaluate their effectiveness later in Section 5. For all attacks, the adversary outputs a guess for the original trace. We start with an attack against the d-privacy-based approach.

4.1 Deobfuscation Attack on d-Privacy

We begin by discussing our threat model, followed by describing the three steps of the attack: (1) Precomputation, (2) Wallet Recovery, and (3) Trace Recovery. In step (2), we first guess the original wallet balance, given an obfuscated wallet balance, and then assign that balance a trace in step (3). Afterward, we discuss what influences the success rate of the deobfuscation attack.

4.1.1 ***Threat Model.*** Although the privacy level in the DP framework is parameterized by ε [10]; it does not reflect the absolute level of privacy for a user, i.e., what can really be inferred from a user's secret [12, 15, 30]. To analyze the privacy level of our mechanism, we consider a threat model where an adversary $\mathcal{A}$ exploits some background information so as to measure what actually can be learned from an obfuscated wallet. Our threat model is similar to the one in [2]. We assume a passive adversary, i.e., it only observes information but does not manipulate any data. $\mathcal{A}$ has access to an obfuscated wallet balance, denoted as w_o, for which it wants to identify the correct (deobfuscated) trace. In addition, it has access to the set of all toll prices P (e.g., by consulting the TSP's website). $\mathcal{A}$ can also obtain all plausible wallets, denoted as W_p, using the toll prices and Eq. (1). Furthermore, $\mathcal{A}$ knows the maximum relative cost that users are expected to pay to maintain privacy, denoted by RE. We assume that RE is publicly available information, so users know what costs to expect. In summary, the adversary's knowledge is represented as $K = \{w_o, P, W_p, RE\}$.

4.1.2 ***Step 1: Precomputation.*** First, $\mathcal{A}$ pre-calculates all possible ranges that an obfuscated balance w_o can fall into. Each range R contains all possible obfuscated balances corresponding to a given original balance w. To create each range, $\mathcal{A}$ computes the lower and upper bounds of the obfuscated values associated with w, using w itself and z. Recall that all generated noises fall within the interval $(-z, z)$ with probability $1 - pr$ (cp. Section 3.1). The pseudocode for the precomputation algorithm is given in Algorithm 2. $\mathcal{A}$ takes W_p, RE as input and computes $z \coloneqq RE \cdot w_{min}$ (cp. Eq. (3)). Then, for each $w \in W_p$, $\mathcal{A}$ computes its lower and upper bounds as $l \coloneqq w - z$ and $u \coloneqq w + z$, respectively, and clamps (l, u) to the interval $[0, w_{max}]$. $\mathcal{A}$ then constructs the corresponding range as $R \coloneqq (l, u)$. Finally, $\mathcal{A}$ stores the pair (R, w) in a list $list\_ranges$.

Note that the precomputation step needs to be executed only once when deobfuscating the first wallet balance and can be skipped when deobfuscating further wallet balances.

4.1.3 ***Step 2: Wallet Recovery.*** The idea behind finding the correct wallet associated with the obfuscated wallet w_o is to first compute which ranges (computed in Step 1) w_o falls into and then select one of them. The goal is to deobfuscate a wallet w_o using the list of precomputed ranges $list\_ranges$ computed in Step 1. For each (R, w) in the list, $\mathcal{A}$ checks whether w_o falls into R or not. If it does, $\mathcal{A}$ retrieves its corresponding wallet w from the tuple (R, w) and adds it to the list of deobfuscated wallets ($list\_deobf\_wallets$) as a candidate for being a deobfuscated wallet. After obtaining the list of deobfuscated wallets, $\mathcal{A}$ selects one of them, denoted as w_c, as its solution for the correct wallet. This selection can be based on different strategies, which will be discussed soon in Section 4.1.4.

Later in the evaluation part, we will need the term *overlapping ranges*: Given two precomputed ranges and their corresponding wallets $\big(R_1 = (w_1 - z, w_1 + z), w_1\big)$ and $\big(R_2 = (w_2 - z, w_2 + z), w_2\big)$, we say that R_1 and R_2 *overlap each other* iff $2 \cdot z > g$ holds, where g is the distance between the wallets w_1 and w_2. Intuitively, deobfuscation becomes more difficult when more ranges overlap. This is because, among the overlapping ranges that include the obfuscated wallet w_o, $\mathcal{A}$ must correctly identify the range that includes the correct wallet w_c corresponding to w_o.

Algorithm 3 Wallet Recovery Attack on d- Privacy

Input: $w_o, list\_ranges$
Output: w_c

```
function METRIC_WALLET_RECOVERY_ATTACK(w_o, list_ranges)
    for all (R, w) ∈ list_ranges do
        if w_o ∈ R then
            list_deobf_wallets ← w
        end if
    end for
    w_c ← SELECT(list_deobf_wallets)
    return w_c
end function
```

4.1.4 ***Success Rate of the Wallet Recovery Attack.*** To obtain the success rate for deobfuscating w_o, we compute the probability that $\mathcal{A}$ correctly guesses the correct trace. We first examine the success rate of the wallet recovery attack.

For the wallet recovery attack, $\mathcal{A}$ is given w_o and has to output the correct original wallet balance w_c, which is among all deobfuscated wallet balances in the list $list\_deobf\_wallets \coloneqq \{w_1, w_2, \ldots, w_k\}$, each of which is a candidate for being a correct wallet. The index c in w_c denotes an arbitrary but fixed wallet in the list $list\_deobf\_wallets$. The adversary could use different strategies to distinguish between the wallets in the list, assigning different probabilities to each as being correct. Consequently, using different strategies could result in different success rates. We now discuss three different strategies.

Baseline Strategy. To select the correct wallet from the list, we treat the following method as a baseline strategy: randomly selecting a wallet (w_i) with equal probability. The success rate of the attack with this strategy, i.e., the probability of correctly guessing the correct original wallet balance, is computed as $SR = 1 / |list\_deobf\_wallets| = 1/k$.

Strategy 1. A more advanced strategy than the baseline strategy is to use the fact that w_is that are closer to w_o are more likely to be connected to w_o. This is due to the fact that smaller deviations (i.e., noise) are more likely than larger ones in the Laplace distribution. To determine the probability that w_i is the correct wallet connected to w_o, the adversary computes the probability ϕ_i for each w_i in the list using Laplace (cp. Definition 2.4):

$$\phi_i \coloneqq \frac{1}{2\lambda} e^{\frac{-|w_o - w_i|}{\lambda}} = \frac{1}{2\lambda} e^{\frac{-|N_i|}{\lambda}}$$

where N_i is the noise added to w_i for obfuscation. Thus, the adversary obtains the set $\{\phi_1, \phi_2, \ldots, \phi_k\}$, where ϕ_i corresponds to w_i in the list of deobfuscated wallets. Since the probabilities in the set might not add up to one, they need to be normalized. Finally, from the list of deobfuscated wallets, the adversary picks the w_i whose associated normalized probability ϕ_i has the highest value. Note that this is a maximum likelihood strategy [33].

Strategy 2. If the adversary has more background knowledge than initially assumed in the threat model (cp. Section 4.1.1) and knows the probabilities of (original) wallet balances occurring in the system, this strategy can be used. Strategy 2 builds upon strategy 1, but the adversary enhances its guessing by utilizing statistics on the frequencies of occurrences of w_i collected over many periods. Basically, the more frequently w_i appears in the system, the more likely it is to be associated with w_o. Let's say the adversary knows the probability, i.e., ω_i, of each plausible wallet w_i occurring in the ETC system. The probability that w_i is connected to w_o is calculated by multiplying two probabilities for two independent events: (1) The probability ω_i that w_i occurs in the ETC system. (2) The probability ϕ_i that w_i is connected to w_o using Laplace (cp. Strategy 1). Thus, the final probability is computed as $\psi_i \coloneqq \phi_i \cdot \omega_i$. Similar to Strategy 1, the adversary obtains the set $\{\psi_1, \psi_2, \ldots, \psi_k\}$, where ψ_i corresponds to w_is and then normalizes the probabilities so that the sum of all ψ_i equals one. Finally, from the list of deobfuscated wallets, the adversary picks the w_i whose associated normalized probability ψ_i has the highest value. Note that we will use Strategy 1 for our evaluation in Section 5.1 since we do not have detailed knowledge of wallet balance occurrences for our case studies.

4.1.5 ***Step 3: Trace Recovery.*** Given a guess of w for the original wallet balance, the adversary now needs to find the trace corresponding to w. For that, we employ the trace finding attack from [2, Section 4], which gets a wallet balance as input and outputs a possible corresponding trace.

The success rate for the complete deobfuscation attack is then determined by multiplying the success rates of the wallet recovery attack with the success rate of the trace recovery attack from [2].

4.2 Deobfuscation Attack on the Exponential Mechanism

We begin by discussing our threat model, followed by describing the two steps of the attack.

4.2.1 ***Threat model.*** Similar to the threat model in Section 4.1.1, we assume a passive adversary $\mathcal{A}$ that has access to the set of all toll prices P and thus also to the set of plausible wallets W_p and the set of plausible traces T_p. Furthermore, we assume for evaluation purposes that $\mathcal{A}$ knows the parameters of the exponential mechanism, i.e., ε, Δ, α_{eucl} and α_{sim}. But depending on who the real-world adversary is, it might not have access to all of these parameters and may have to guess (some of) them. It also knows an obfuscated trace $trace_o$ which it wants to deobfuscate. In summary, our adversary knowledge is represented as $K = \{trace_o, P, W_p, T_p, \varepsilon, \Delta, \alpha_{eucl}, \alpha_{sim}\}$.

4.2.2 ***Deobfuscation Attack.*** The attack includes two steps: (1) precomputation and (2) deobfuscation. In the precomputation phase (cp. Algorithm 4), $\mathcal{A}$ creates a table containing the probabilities that $trace_i$ got mapped to $trace_j$ during obfuscation, for $i, j \in \{0, \ldots, |T_p| - 1\}$. For that, the exponential *ob*fuscation algorithm (Algorithm 1) is executed for each $trace_i \in T_p$ to get the probability that this trace gets mapped to $trace_j$, for all $trace_j \in T_p$. The results are stored in a table, where the cell (i, j) contains the likelihood that $trace_i$ is mapped to $trace_j$. .

In the deobfuscation phase (cp. Algorithm 5), given an obfuscated trace $trace_j$, $\mathcal{A}$ simply selects the cell in the jth column that contains the maximum of the jth column. This cell holds the id of the trace with the highest probability of being the original trace. Note that attack is again a maximum likelihood strategy.

Algorithm 4 Precomputation for Attack on the Exponential Mechanism

Input: $T_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}$
Output: $table$

function PRECOMPUTATION_EXP_ATTACK($T_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}$)
  Declare $table[|T_p|][|T_p|]$ \\Create 2-dimensional array
  for all $trace_i \in T_p$ **do**
    $input \leftarrow (trace_i, T_p, \varepsilon, \alpha_{eucl}, \alpha_{sim})$
    $arr\_norm\_prob \leftarrow$ COMPUTE_NORMALIZED_PROB($input$)
    \\This executes EXPONENTIAL_OBFUSCATION until arr_norm_prob is computed
    $table[i] \leftarrow arr\_norm\_prob$
  end for
  return $table$
end function

Algorithm 5 Attack on the Exponential Mechanism

Input: $trace_o \in T_p, T_p, table$
Output: $trace_c$

function EXP_ATTACK($trace_o, T_p, table$)
  Declare $column[|T_p|]$
  Let j be the index for which $trace_o = T_p[j]$ holds
  for i from 0 to $|T_p| - 1$ **do**
    $column[i] \leftarrow table[i][j]$ \\get jth column of $table$
  end for
  $index \leftarrow arg\_max(column)$ \\Find maximum probability
  $trace_c \leftarrow T_p[index]$
  return $trace_c$
end function

5 EVALUATION

We now evaluate the effectiveness of our deobfuscation attacks against our wallet obfuscation mechanisms to determine the level of privacy achievable and the associated costs. Using the current ETC systems in Brisbane and Melbourne as case studies, we apply their parameters,i.e., toll stations and prices, to a hypothetical PPETC scheme. We then assess whether adding d-privacy or exponential-DP to this scheme helps in hiding the toll station visits. The adversary's goal, in this section, is to recover the original trace of the user, given an obfuscated wallet. Note that in Appendix G, we consider a more relaxed attacker where the adversary tries to find a trace that is just similar to the original trace instead of finding the exact trace.

5.1 Evaluation of d-Privacy

We analyze the effectiveness of the deobfuscation attack against the d-privacy approach as follows.

Privacy Analysis. We evaluate the privacy level of an individual by calculating the success rate of the deobfuscation attack from Section 4.1, while using Strategy 1 (cp. Section 4.1.4) for the selection of the deobfuscated wallet, for different ε, namely $\varepsilon \in \{0.5, 1, 5\}$. More precisely, we perform the following steps:

Step 1: For a given $\varepsilon \in \{0.5, 1, 5\}$, we compute the corresponding relative error bound RE and the noise bound z. We then execute the precomputation phase (cp. Section 4.1.2) to get for each wallet w the range R in which all obfuscated wallets fall into (with probability $1 - pr$).

Step 2: For each wallet w, we take the corresponding range R of possible obfuscated wallets (computed in step 1). Then, for each $w_o \in R$ we execute the wallet recovery attack (cp. Section 4.1.3).

Step 3: To get the results for the complete deobfuscation attack, we multiply the probability that the adversary can recover the correct wallet *balance* (obtained in step 2) with the probability that the correct *trace* can be recovered from that wallet balance.

For each $\varepsilon \in \{0.5, 1, 5\}$, we provide the results of the privacy analysis for the complete deobfuscation attack as a graph. Note that in Appendix E we additionally provide the results for just the wallet recovery attack (without trace recovery).

Cost Analysis. We also evaluate the amount of additional noise introduced by obfuscation for the same εs used in the privacy evaluation. This helps us to understand at what cost the level of privacy is achieved. The cost analysis includes the following steps:

Step 1: For different ε and its associated RE, we add noise to each plausible wallet w to obtain its associated obfuscated wallet w_o, using Algorithm 6.

Step 2: Having obtained an obfuscated wallet, we compute the cost as $|w_o - w|$, for each w and its corresponding w_o.

We repeat this 1000 times to get a good estimate of the costs. The cost analysis results are presented in a table, where each row displays various parameters, including pr, RE, ε, non-outliers, outliers, and z. The columns for non-outliers and outliers are determined using a box plot method that identifies noise data distribution (explained in Appendix C.2). The non-outliers, outliers, and z, are given in dollars.

5.1.1 *Brisbane Case Study.*

Parameters. To evaluate our attack, we utilize the actual parameters of Brisbane's ETC system [6, 34, 35], which has also been examined in [2]. We use the following parameters for our evaluation:

Toll prices (P): The 9 toll prices (in dollars) are as follows [34]: $P = \{1.72, 2.68, 2.84, 3.19, 4.09, 4.55, 5.11, 5.11, 5.46\}$.

Plausible wallets (W_p): Based on the toll prices, we can compute the set of plausible wallets W_p within the range $[\$0, \$10]$[5] using Eq. (1).

Obfuscated wallets (W_o): We obtain the ranges that include the obfuscated wallets through the precomputation phase.[6] Parameters needed for obfuscation of wallets include RE, ε, w_{min}, pr, and Δ. The parameter RE is computed using $\varepsilon \in \{0.5, 1, 5\}$, $\Delta = 1$, $w_{min} = 1.72$ and $pr = 0.001$. Note that the parameters ε, and pr are not available to the adversary; they are only used for the purpose of our evaluation.

[5] Since we will see that larger wallets are easier to obfuscate, we intentionally examine only "small" wallets to better see the effects of the obfuscation.
[6] Since it is not feasible to display every precomputed range on the x-axis of the graphs, we exclude ranges with success rates that are very similar.

pr	RE	ε	non-outliers	outliers	z
0.001	8.0	0.5	(-4.0, 4.6)	(-8.2, 9.9)	13.82
0.001	4.0	1.0	(-2.2, 2.4)	(-7.3, 9.5)	6.91
0.001	0.8	5.0	(-0.5, 0.5)	(-2.6, 2.0)	1.38

Table 1: d-privacy for Brisbane case study: Each row shows the noise range w.r.t different ε and corresponding RE.

pr	RE	ε	non-outliers	outliers	z
0.001	7.2	0.5	(-4.1, 4.7)	(-8.1, 10.0)	13.82
0.001	3.6	1.0	(-2.4, 2.6)	(-6.9, 9.2)	6.91
0.001	0.7	5.0	(-0.5, 0.5)	(-2.0, 1.5)	1.38

Table 2: d-privacy for Melbourne case study: Each row shows the noise range w.r.t different ε and corresponding RE.

Privacy Analysis. Figure 1a shows the success rate of the *complete deobfuscation attack*. Each range $(\cdot, \cdot)$ on the x-axis encompasses all obfuscated wallets w_o w.r.t one plausible wallet w. Note that for space reasons, not all plausible wallets are labeled on the x-axis, but all 93 possible wallets in the range [\$0, \$10] have their success chance plotted. Overall, Fig. 1a demonstrates that for $\varepsilon \in \{0.5, 1\}$ all success rates are pretty similar. Only the smallest and the second-smallest wallet balances have a deobfuscation success rate >5%, and most wallet balances have a deobfuscation success rate <0.2%. For $\varepsilon = 5$, only wallet balances below \$6.77 have a deobfuscation success rate >5%. Note that success rates are higher than the corresponding ones in the graphs for $\varepsilon \in \{0.5, 1\}$. Since here considerably less noise was used, this is not surprising.

In all three graphs it is evident that higher wallet balances have lower success rates. This is because there are significantly more possible wallets with higher balances than ones with lower balances, and thus higher balances are harder to deobfuscate due to there being more possible original balances.

Cost Analysis. Each row in Table 1 displays the non-outlier and outlier noise values corresponding to $\varepsilon \in \{0.5, 1, 5\}$. The table clearly shows the trade-off between privacy and cost. $\varepsilon = 0.5$ provides very good privacy, but the expected cost (non-outlier) is quite high at <5 dollars. For $\varepsilon = 1$, which also achieves very good results in the privacy analysis, the expected costs are significantly better at <2.6 dollars, but are still above w_{min} (1.72 dollars). Only for $\varepsilon = 5$ are the expected costs <0.6 dollars, but this variant performs notably worse than the other two in the privacy analysis.

Remark 3. Note that approximately 94% of the noise values are non-outliers, which satisfies the threshold defined by RE and are in the range $(-z, z)$. About 6% of noises are considered outliers, some of which are greater than the noise bound z. The probability that a generated noise is outside $(-z, z)$ is set to $pr = 0.001$.

5.1.2 *Melbourne Case Study.*

Parameters. As a second real-world example, we examine a PPETC system based on the Melbourne ETC system [26], which has the following parameters:

- **Toll prices (P):** We assume the following 19 toll prices (in dollars): $P = \{1.92, 1.92, 3.07, 3.07, 3.07, 3.07, 3.84, 3.84, 4.99, 6.14, 6.14, 6.91, 6.91, 6.91, 8.06, 9.98, 9.98, 9.98, 10.75\}$.
- **Plausible wallets (W_p):** As for the Brisbane case study, we obtain all plausible wallet balances within the range [\$1, \$10] with Eq. (1).
- **Obfuscated wallets (W_o):** As for the Brisbane case study, the parameters needed for the obfuscation of wallets include RE, ε, w_{min}, pr, and Δ. The parameter RE is computed using $\varepsilon \in \{0.5, 1, 5\}$, $\Delta = 1$, $w_{min} = 1.92$ and pr, which we set to $pr = 0.001$.

Privacy Analysis. For the *complete deobfuscation attack*, Fig. 1b shows a trend similar to Fig. 1a, but with a notable difference: the overall success rate is smaller compared to the Brisbane case study. For $\varepsilon \in \{0.5, 1\}$, the difference is that now only the smallest wallet balances have a success rate >5%. A more notable difference is for $\varepsilon = 5$, where the success rate is >5% for the three smallest wallet balances and pretty small for all others.

When comparing the results from the complete attack with only the wallet recovery attack (cp. Fig. 4b), the complete deobfuscation attack performs notably better than the wallet recovery attack alone. This is probably due to the fact that our price list for Melbourne contains several toll stations that share the same price, thus making visits to them indistinguishable given a wallet balance.

Remark 4. Figure 1b shows only 13 different wallet ranges since there are only 13 possible wallet balances that are ≤10 dollars.

Cost Analysis. Table 2 is very similar to Table 1, since they share the same z values (cp. Eq. (3)). The only difference is that the relative error bound RE is smaller than for Brisbane since w_{min} is higher.

5.2 Evaluation of the Exponential Mechanism

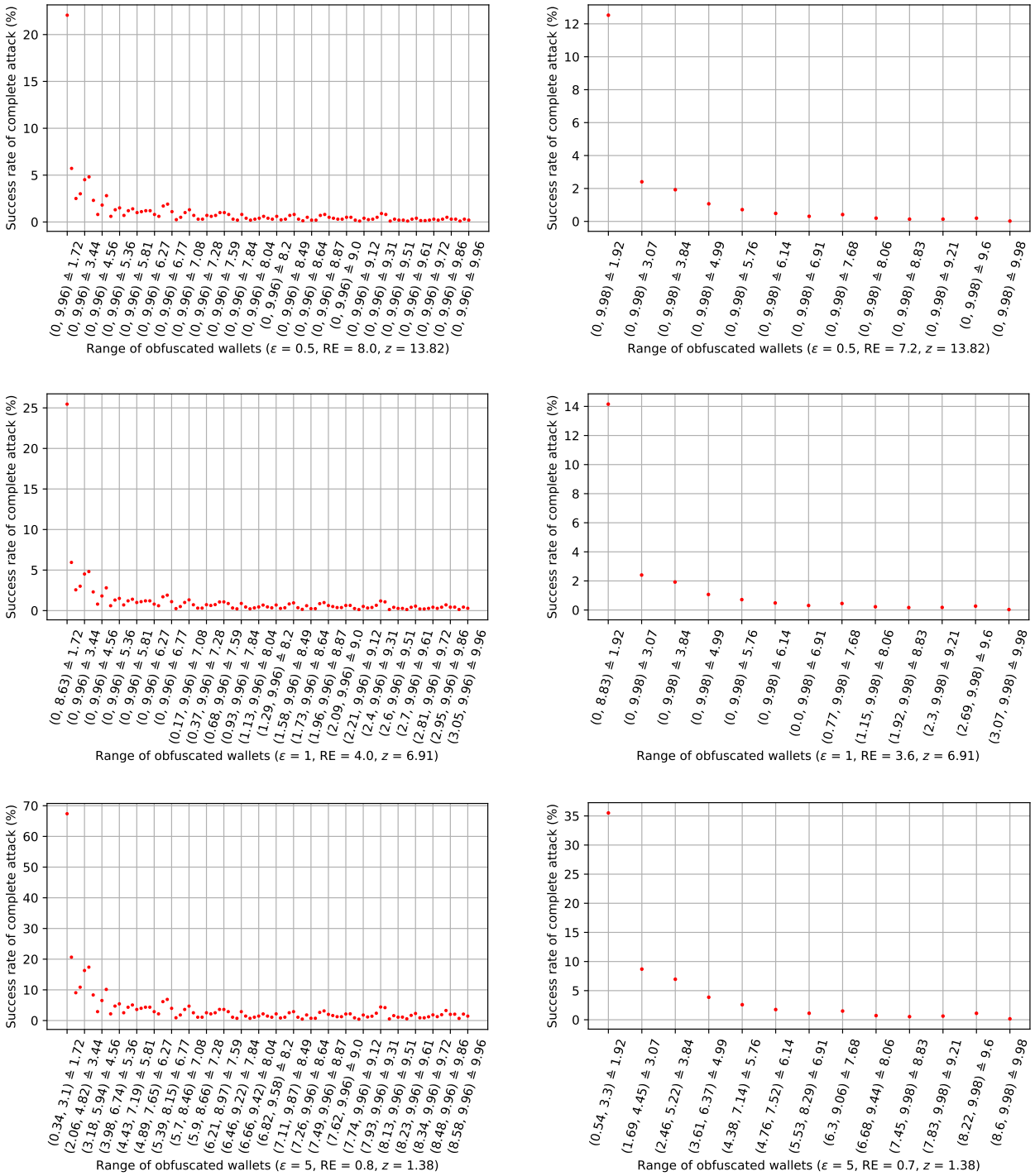
We analyze the effectiveness of the deobfuscation attack against the exponential mechanism as follows.

Privacy Analysis. We evaluate the privacy level of an individual by calculating the success rate of the deobfuscation attack from Section 4.2 for the selection of the deobfuscated trace, for different ε. To do so, for each obfuscated trace from the set T_p, we compute the success rate of finding the associated origin trace from the set T_p. The results of the privacy analysis are presented in a series of graphs, where each graph shows the success rate with respect to an obfuscated trace, for different ε, α_{eucl}, and α_{sim}.

For ε, we will analyze $\varepsilon \in \{0.5, 1, 5\}$. Due to space restrictions, we will focus on $(\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25)$ in this section. In Appendix F, we will also evaluate for the parameters $(\alpha_{eucl} = 1, \alpha_{sim} = 0)$ and $(\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5)$.

Cost Analysis. We evaluate the amount of additional noise introduced by obfuscation for the different ε used in the privacy analysis. This helps us to understand at what cost the level of privacy is achieved. The cost analysis includes the following steps:

- **Step 1:** For a fixed ε and for each plausible trace $trace$, we obtain its associated obfuscated trace $trace_o$, using the exponential obfuscation mechanism (Algorithm 1).



(a) Brisbane **(b) Melbourne**

Figure 1: d-privacy: Each graph shows the success rate of the *complete deobfuscation attack* w.r.t a certain ε and corresponding RE, z. On the x-axis, each range (in dollars) includes all possible obfuscated wallets (w_o), linked with a certain original wallet (w). For example, the first x-axis value on the lower left graph (Brisbane with $\varepsilon = 5$) depicts the wallet with balance $1.72. For this wallet, the obfuscated wallet balances fall into the range $[\$0.34, \$3.1]$. On the y-axis, the success rate of the attack is shown, i.e., the percentage of successful deobfuscations. (Brisbane and Melbourne case study)

Step 2: Having obtained the obfuscated trace, we compute the cost as $|\text{GET\_BALANCE}(trace_o) - \text{GET\_BALANCE}(trace)|$.

We repeat this 1000 times to get a good estimate of the costs. The cost analysis results are presented in a table, where each row displays parameters, including wallet (consisting of wallet id and wallet balance), ε, non-outliers, and outliers. The character N in the tables (if present) indicates that no outliers occurred in the corresponding row.

5.2.1 ***Brisbane Case Study.*** We use the same parameters as in Section 5.1.1.

Privacy Analysis. Figure 2a shows the success rate of the deobfuscation attack on the exponential mechanism for Brisbane. In contrast to d-privacy (cp. Fig. 1a), for $\varepsilon \in \{0.5, 1\}$ *all* success rates are between 0.9% and 1.2%, thus very close together. For $\varepsilon = 5$, the seven smallest wallets have success rates between 2% and 2.8%, and all other wallets have a success rate <2%. Overall, the exponential mechanism performs significantly better than d-privacy for *small* wallet balances, but for larger balances, both perform similarly (they are all <4% and thus all relatively small). It is noticeable that for the exponential mechanism, the success rates for different balances are all very close together, whereas they fluctuate more for d-privacy.

Cost Analysis. In Table 3 we show the cost of privacy for $\varepsilon \in \{0.5, 1, 5\}$. It should be noted that, compared to d-privacy, where each wallet had the same noise bound, the imposed noise for the exponential mechanism differs from wallet to wallet. The imposed noise for the exponential mechanism also seems to generally be higher than for the d-privacy mechanism (cp. Table 1). For $\varepsilon = 0.5$, the difference is not that significant, but for $\varepsilon \in \{1, 5\}$ the imposed noise for exponential is considerably higher than the expected noise for d-privacy. It is noticeable that the first wallet has by far the highest cost. This is due to the fact that this wallet is the smallest and can thus only be mapped to wallets with a greater (or equal) balance, whereas other wallets can be mapped to wallets with a greater or smaller balance. Combined with the fact that the noise is depicted as the *absolute* difference between balances, it is not surprising that the smallest wallet has a higher cost than the others.

5.2.2 ***Melbourne Case Study.*** We use the same parameters as in Section 5.1.2.

Privacy Analysis. Figure 2b shows the success rate of the deobfuscation attack on the exponential mechanism for Melbourne. The same observations that were made for Brisbane (cp. Fig. 2a) can be made for Melbourne as well, with the difference that all graphs have even lower success chances than the exponential mechanism for Brisbane, for $\varepsilon \in \{0.5, 1\}$ all are <0.5% and for $\varepsilon = 5$ all are <1.5%. This is probably due to the fact that the set T_p for Melbourne is more than two times as large as the one for Brisbane, thus the set of possible deobfuscated traces is much larger.

Cost Analysis. The costs for Melbourne under the exponential mechanism (depicted in Table 4) are still higher than those under d-privacy (cp. Table 2), but they are significantly lower than for Brisbane under the exponential mechanism (cp. Table 3). The fact that traces from Melbourne are apparently easier to obfuscate than traces from Brisbane is consistent with the results from the privacy analysis of the exponential mechanism, where traces from Melbourne were more difficult to deobfuscate.

6 DISCUSSION

Discussion of Evaluation Results. The *privacy analysis* of the Brisbane and Melbourne ETC schemes, for both d-privacy and the exponential mechanism, shows that the success rate differs between smaller and larger wallet balances. It is noticeable that larger wallet balances provide significantly more privacy than smaller ones since for higher wallet balances, there are more distinct traces that have the same wallet balance. It is noticeable that the smallest wallet balance (w_{min}) has by far the highest success rate in all scenarios. This is due to the fact that there are by far more traces that have a high wallet balances than ones with a small wallet balance, thus making traces with a high balance easier to obfuscate.

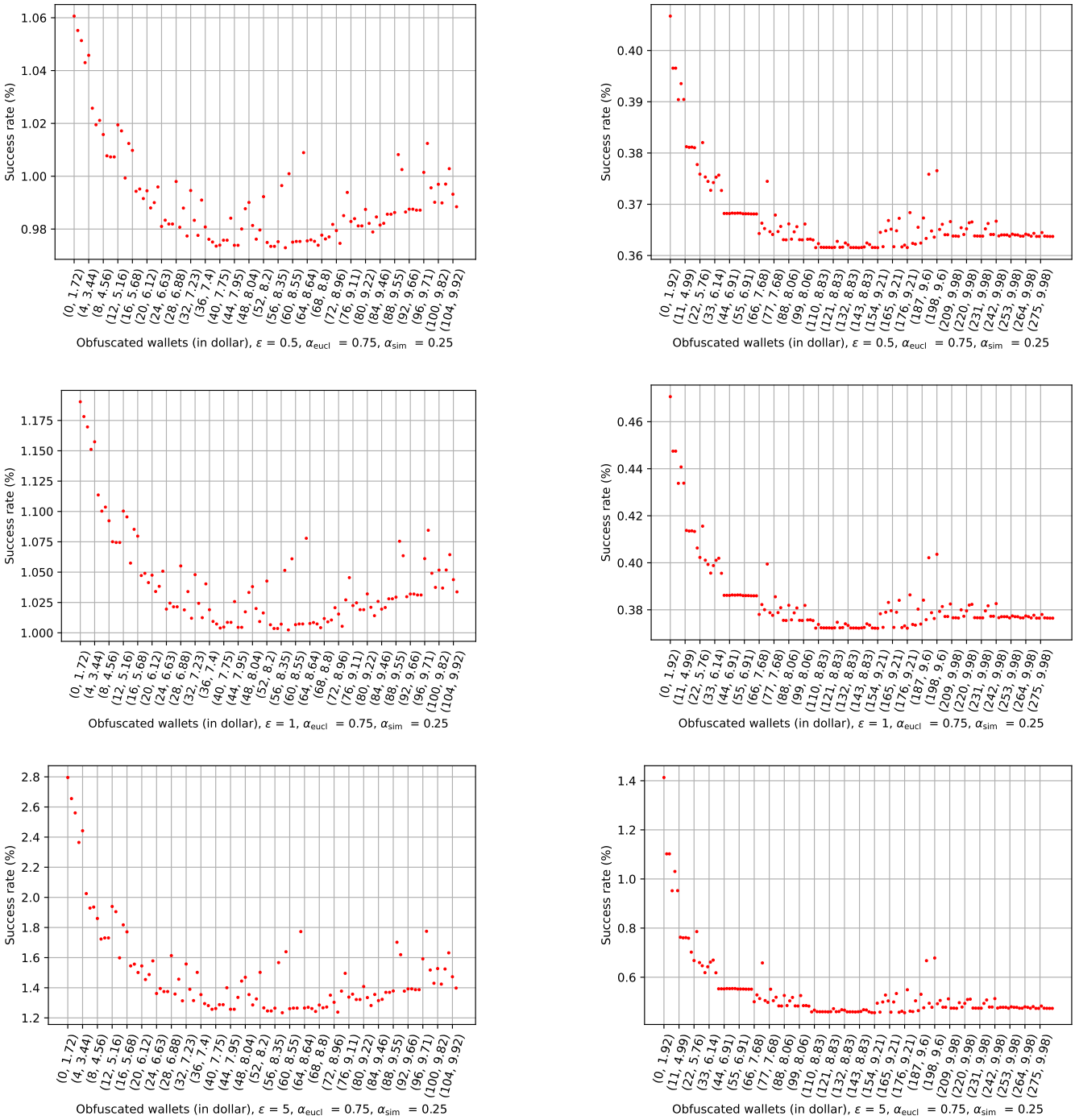
When comparing d-privacy to the exponential mechanism, it can be deducted from Figs. 1 and 2 that, for all ε, the exponential mechanism performs significantly better for small wallets. For larger wallets, while both perform similar *on average*, the exponential mechanism consistently has very low success rates, while the success rates for d-privacy depend more on the individual balance and thus fluctuate more. Thus, if success rates need to be *consistently* small, using the exponential mechanism would be a safer option.

When comparing the *cost* for d-privacy and the exponential mechanism (cp. Tables 1, 2, 3 and 4), it can be seen that the exponential mechanism needs more noise than the d-privacy mechanism.

In conclusion, while the exponential mechanism yields good privacy for *all* traces, it also has a higher cost. It is thus up to the TSP to select a suitable mechanism, depending on how they want to balance user privacy and cost.

Reimbursement of Additional Costs. When using the ETC scheme for many billing periods, the cost for privacy equals the sum of the individual noises, i.e., $C \coloneqq \sum_i N_i$. The expectation value for C after a great number of billing periods is close to zero. Thus, if the ETC is used for a long time, the *actual* additional cost of privacy should be small for most users. Of course, there will always be users who are a bit unlucky and may end up with a large additional cost. Thus, the TSP could offer some kind of *reimbursement mechanism*, like the one in [10], for the excess cost.

Further Research Opportunities. Note that our evaluation only considers *one billing period*. By consolidating information about multiple billing periods, the adversary may have a higher chance of violating the user's privacy. Intuitively, the adversary's chances of doing so depend on the behavior of users. For example, consider the extreme case of a user who has exactly the same trace and, thus, the same wallet balance every month. The adversary could analyze the obfuscated wallets over several months and deduce the correct original traces with a higher probability than if only one month was considered. For example, for d-privacy, the adversary could divide the sum of obfuscated balances by the number of billing periods, which gives a good estimate of the real balances (if the number of billing periods is large enough). Conversely, for users whose driving behavior changes significantly from month to month, their obfuscated wallets would be harder to deobfuscate



(a) Brisbane **(b) Melbourne**

Figure 2: Exponential mechanism: Each graph shows the success rate of deobfuscation for $\varepsilon \in \{0.5, 1, 5\}$ and for ($\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25$). The tuples on the x-axis indicate $(id, wallet)$ where id distinguishes identical wallets with different traces. (Brisbane and Melbourne case study)

ε	(id, wallet)	non_outliers	outliers
0.5	(0, 1.72)	(1.7, 8.2)	(0.0, 1.5)
0.5	(10, 5.11)	(0.0, 4.9)	N
0.5	(20, 6.12)	(0.0, 4.4)	N
0.5	(30, 7.08)	(0.0, 4.4)	(5.4, 5.4)
0.5	(40, 7.75)	(0.0, 3.7)	(4.3, 6.0)
0.5	(50, 8.14)	(0.0, 3.6)	(3.7, 6.4)
0.5	(60, 8.55)	(0.0, 2.9)	(3.0, 6.8)
0.5	(70, 8.9)	(0.0, 4.0)	(4.3, 7.2)
0.5	(80, 9.22)	(0.0, 5.1)	(5.8, 7.5)
0.5	(90, 9.57)	(0.0, 6.1)	(6.4, 7.9)
0.5	(100, 9.82)	(0.0, 6.4)	(6.6, 8.1)

(a) $\varepsilon = 0.5$

ε	(id, wallet)	non_outliers	outliers
1	(0, 1.72)	(1.0, 8.2)	(0.0, 0.0)
1	(10, 5.11)	(0.0, 4.9)	N
1	(20, 6.12)	(0.0, 4.4)	N
1	(30, 7.08)	(0.0, 4.4)	(5.4, 5.4)
1	(40, 7.75)	(0.0, 3.7)	(4.3, 6.0)
1	(50, 8.14)	(0.0, 3.2)	(3.6, 6.4)
1	(60, 8.55)	(0.0, 2.9)	(3.0, 6.8)
1	(70, 8.9)	(0.0, 4.0)	(4.3, 7.2)
1	(80, 9.22)	(0.0, 4.3)	(4.7, 7.5)
1	(90, 9.57)	(0.0, 5.5)	(6.1, 7.9)
1	(100, 9.82)	(0.0, 5.7)	(6.4, 8.1)

(b) $\varepsilon = 1$

ε	(id, wallet)	non_outliers	outliers
5	(0, 1.72)	(0.0, 8.2)	N
5	(10, 5.11)	(0.0, 4.9)	N
5	(20, 6.12)	(0.0, 4.4)	N
5	(30, 7.08)	(0.0, 4.2)	(4.4, 5.4)
5	(40, 7.75)	(0.0, 3.7)	(4.3, 6.0)
5	(50, 8.14)	(0.0, 3.0)	(3.2, 6.4)
5	(60, 8.55)	(0.0, 2.5)	(2.7, 6.8)
5	(70, 8.9)	(0.0, 2.3)	(2.5, 7.2)
5	(80, 9.22)	(0.0, 3.0)	(3.1, 7.5)
5	(90, 9.57)	(0.0, 4.4)	(4.5, 7.9)
5	(100, 9.82)	(0.0, 4.7)	(4.9, 8.1)

(c) $\varepsilon = 5$

Table 3: Exponential mechanism for Brisbane case study: Each row shows the cost (in dollars) w.r.t a wallet (associated with a trace) and a fixed ε, using ($\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25$). The id in $(id, wallet)$ distinguishes identical wallets with different traces.

ε	(id, wallet)	non_outliers	outliers
0.5	(0, 1.92)	(1.9, 8.1)	(0.0, 1.1)
0.5	(26, 5.76)	(0.0, 4.2)	N
0.5	(52, 6.91)	(0.0, 5.0)	N
0.5	(78, 7.68)	(0.0, 3.8)	(4.6, 5.8)
0.5	(104, 8.06)	(0.0, 3.1)	(4.2, 6.1)
0.5	(130, 8.83)	(0.0, 1.9)	(2.7, 6.9)
0.5	(156, 9.21)	(0.0, 3.1)	(3.5, 7.3)
0.5	(182, 9.6)	(0.0, 3.8)	(4.6, 7.7)
0.5	(208, 9.98)	(0.0, 5.0)	(6.1, 8.1)
0.5	(234, 9.98)	(0.0, 5.0)	(6.1, 8.1)
0.5	(260, 9.98)	(0.0, 4.2)	(5.0, 8.1)

(a) $\varepsilon = 0.5$

ε	(id, wallet)	non_outliers	outliers
1	(0, 1.92)	(1.1, 8.1)	(0.0, 0.0)
1	(26, 5.76)	(0.0, 4.2)	N
1	(52, 6.91)	(0.0, 5.0)	N
1	(78, 7.68)	(0.0, 3.8)	(4.6, 5.8)
1	(104, 8.06)	(0.0, 3.1)	(4.2, 6.1)
1	(130, 8.83)	(0.0, 1.9)	(2.7, 6.9)
1	(156, 9.21)	(0.0, 3.1)	(3.5, 7.3)
1	(182, 9.6)	(0.0, 2.7)	(3.5, 7.7)
1	(208, 9.98)	(0.0, 4.2)	(5.0, 8.1)
1	(234, 9.98)	(0.0, 4.2)	(5.0, 8.1)
1	(260, 9.98)	(0.0, 5.0)	(6.1, 8.1)

(b) $\varepsilon = 1$

ε	(id, wallet)	non_outliers	outliers
5	(0, 1.92)	(0.0, 8.1)	N
5	(26, 5.76)	(0.0, 4.2)	N
5	(52, 6.91)	(0.0, 5.0)	N
5	(78, 7.68)	(0.0, 3.8)	(4.6, 5.8)
5	(104, 8.06)	(0.0, 3.1)	(4.2, 6.1)
5	(130, 8.83)	(0.0, 1.9)	(2.7, 6.9)
5	(156, 9.21)	(0.0, 2.3)	(3.1, 6.1)
5	(182, 9.6)	(0.0, 2.7)	(3.5, 7.7)
5	(208, 9.98)	(0.0, 2.3)	(3.1, 8.1)
5	(234, 9.98)	(0.0, 2.3)	(3.1, 8.1)
5	(260, 9.98)	(0.0, 4.2)	(5.0, 8.1)

(c) $\varepsilon = 5$

Table 4: Exponential mechanism for Melbourne case study: Each row shows the cost (in dollars) w.r.t a wallet (associated with a trace) and a fixed ε, using ($\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25$). The id in $(id, wallet)$ distinguishes identical wallets with different traces.

successfully. Understanding how user behavior impacts privacy and determining how varied a user's activities need to be to prevent privacy loss are important areas for future research.

7 RELATED WORK

Attacks on PPETC Schemes. Attacks on post-payment PPETC schemes have been considered in [2] and [8]. Both use knowledge of wallet balances and try to solve the SSP to obtain additional information about user behavior. In [2] the evaluation is based on real ETC data. It is shown that the Brisbane scenario from Section 5.1.1 is vulnerable to attacks: For wallet balances ≤10 dollars, the adversary could identify the visited toll stations with a 94% success rate by solving the SSP problem. [8] also shows that solving the SSP helps to effectively recover user traces. In contrast to [2], the adversary in [8] uses every information the TSP gets.

Protection Mechanisms. Some PPETC schemes [4, 18, 29] briefly mention that solving the SSP might lead to privacy problems, although no solutions are presented. We are only aware of one work that examines possible protection mechanisms: While [10] does not directly look at PPETC schemes, they consider the more general setting of applications that use fine-grained billing, where the details of the billing are hidden from the service provider. Their central idea is to use the DP framework to add noise to the final bill of a user. The noise can be freely chosen by the user, which in practice may lead to the problem that most users will choose a noise of zero to save costs and thus get no privacy gain. [10] additionally proposes a cryptographic protocol that helps customers reclaim the additional expenditure incurred for the sake of privacy. A limitation of [10] is that it does not consider protection against adversaries who may exploit background information. In contrast, we present attacks and use real-world settings from the Brisbane and Melbourne ETC systems to evaluate our mechanism against adversaries who may exploit background information.

In smart metering applications, rather than transmitting actual measurements, it is possible that the smart meter sends masked data to the power provider in a way that does not interfere with the accuracy of aggregation operations. [5] and [19] present methods to obscure measurements using a straightforward approach. Specifically, the smart meter adds noise from a Laplace distribution

with a certain scale parameter and transmits this data to the power provider. The scale parameter is selected to ensure that the cumulative noise remains below a predefined error threshold. The authors explain that when a large number of measurements is considered, this cumulative error approximates a normal distribution. However, their method is not suitable for our scenario, as the number of obfuscated wallets may be too small for their approach to work effectively. Additionally, their method does not consider controlling noise for individual measurements.

8 CONCLUSION

Since previous work has shown that a PPETC version of the Brisbane ETC system is vulnerable to trace recovery attacks (success rates up to 94%), we have investigated in this work whether common ε-DP mechanisms such as d-privacy or the exponential mechanism could help to prevent these attacks. Our analysis showed that both mechanisms are very good at making these attacks more difficult. But privacy does not come for free: basically, users must be willing to pay a slightly randomized price for using the ETC system. Although the expected cost of privacy is small over a large number of billing periods, the cost for a single billing period can be several dollars for $\varepsilon \leq 1$, which may discourage users from using the ETC system. It is up to the TSP to balance the cost that users are willing to pay against the privacy that can be achieved.

ACKNOWLEDGMENTS

The authors used ChatGPT-3.5 to correct some typos, grammatical errors, and awkward phrasing.

REFERENCES

[1] Alessandro Acquisti, Leslie K John, and George Loewenstein. 2013. What is privacy worth? *The Journal of Legal Studies* 42, 2 (2013), 249–274.
[2] Amirhossein Adavoudi Jolfaei, Andy Rupp, Stefan Schiffner, and Thomas Engel. 2024. Why Privacy-Preserving Protocols Are Sometimes Not Enough: A Case Study of the Brisbane Toll Collection Infrastructure. *Proceedings on Privacy Enhancing Technologies* 2024, 1 (2024).
[3] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2013. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 901–914.
[4] Josep Balasch, Alfredo Rial, Carmela Troncoso, Bart Preneel, Ingrid Verbauwhede, and Christophe Geuens. 2010. PrETP:Privacy-Preserving Electronic Toll Pricing. In *19th USENIX Security Symposium (USENIX Security 10)*.
[5] Pedro Barbosa, Andrey Brito, and Hyggo Almeida. 2016. A technique to provide differential privacy for appliance usage in smart metering. *Information Sciences* 370 (2016), 355–367.
[6] Linkt Brisbane. 2022. *Queensland toll calculator.* Retrieved 2022 from https://www.linkt.com.au/using-toll-roads/toll-calculator/brisbane
[7] Konstantinos Chatzikokolakis, Miguel E Andrés, Nicolás Emilio Bordenabe, and Catuscia Palamidessi. 2013. Broadening the scope of differential privacy using metrics. In *Privacy Enhancing Technologies: 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings 13*. Springer, 82–102.
[8] Xihui Chen, David Fonkwe, and Jun Pang. 2012. Post-hoc user traceability analysis in electronic toll pricing systems. In *Data Privacy Management and Autonomous Spontaneous Security*. Springer, 29–42.
[9] Dan Cvrcek, Marek Kumpost, Vashek Matyas, and George Danezis. 2006. A study on the value of location privacy. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*. 109–118.
[10] George Danezis, Markulf Kohlweiss, and Alfredo Rial. 2011. Differentially private billing with rebates. In *Information Hiding: 13th International Conference, IH 2011, Prague, Czech Republic, May 18-20, 2011, Revised Selected Papers 13*. Springer, 148–162.
[11] Cynthia Dwork. 2006. Differential privacy. In *International colloquium on automata, languages, and programming*. Springer, 1–12.
[12] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*. Springer, 1–19.
[13] Cynthia Dwork, Nitin Kohli, and Deirdre Mulligan. 2019. Differential privacy in practice: Expose your epsilons! *Journal of Privacy and Confidentiality* 9, 2 (2019).
[14] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer, 265–284.
[15] Cynthia Dwork and Moni Naor. 2010. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality* 2, 1 (2010).
[16] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.
[17] Natasha Fernandes. 2021. *Differential privacy for metric spaces: information-theoretic models for privacy and utility with new applications to metric domains*. Ph. D. Dissertation. École Polytechnique Paris; Macquarie University.
[18] Valerie Fetzer, Max Hoffmann, Matthias Nagel, Andy Rupp, and Rebecca Schwerdt. 2020. P4TC—Provably-Secure yet Practical Privacy-Preserving Toll Collection. *Proceedings on Privacy Enhancing Technologies* (2020).
[19] Matthew Hale, Prabir Barooah, Kendall Parker, and Kasra Yazdani. 2019. Differentially private smart metering: Implementation, analytics, and billing. In *Proceedings of the 1st ACM International Workshop on Urban Building Energy Sensing, Controls, Big Data Analysis, and Visualization*. 33–42.
[20] Il-Horn Hann, Kai-Lung Hui, Sang-Yong Tom Lee, and Ivan PL Png. 2007. Overcoming online information privacy concerns: An information-processing theory approach. *Journal of management information systems* 24, 2 (2007), 13–42.
[21] Amirhossein Adavoudi Jolfaei, Abdelwahab Boualouache, Andy Rupp, Stefan Schiffner, and Thomas Engel. 2023. A Survey on Privacy-Preserving Electronic Toll Collection Schemes for Intelligent Transportation Systems. *IEEE Transactions on Intelligent Transportation Systems* (2023).
[22] Frank Kargl, Arik Friedman, and Roksana Boreli. 2013. Differential privacy in intelligent transportation systems. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*. 107–112.
[23] Florian Kerschbaum and Hoon Wei Lim. 2015. Privacy-preserving observation in public spaces. In *European Symposium on Research in Computer Security*. Springer, 81–100.
[24] Masanobu Kii, Atsunori Ichikawa, and Takayuki Miura. 2025. Lightweight Two-Party Secure Sampling Protocol for Differential Privacy. *Proceedings on Privacy Enhancing Technologies* (2025), 23—-36. Issue 1. https://doi.org/10.56553/popets-2025-0003
[25] Jeffrey C Lagarias and Andrew M Odlyzko. 1985. Solving low-density subset sum problems. *Journal of the ACM (JACM)* 32, 1 (1985), 229–246.
[26] Linkt. 2024. *Toll pricing.* Retrieved 2024 from https://www.linkt.com.au/using-toll-roads/about-toll-roads/citylink/toll-pricing/melbourne
[27] Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 94–103.
[28] KW Ogden. 2001. Privacy issues in electronic toll collection. *Transportation Research Part C: Emerging Technologies* 9, 2 (2001), 123–134.
[29] Raluca Ada Popa, Hari Balakrishnan, and Andrew J Blumberg. 2009. VPriv: Protecting privacy in location-based vehicular services. (2009).
[30] Reza Shokri. 2014. Privacy games: Optimal user-centric data obfuscation. *arXiv preprint arXiv:1402.3426* (2014).
[31] Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. 2011. Quantifying location privacy. In *2011 IEEE symposium on security and privacy*. IEEE, 247–262.
[32] Edward Skidelsky and Robert Skidelsky. 2012. *How much is enough?: money and the good life*. Penguin UK.
[33] George Theodorakopoulos. 2015. The same-origin attack against location privacy. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*. 49–53.
[34] Transport and Public Works Committee. 2018. *Inquiry into the operations of toll roads in Queensland.* Retrieved 2022 from https://www.transurban.com/content/dam/transurban-pdfs/02/news/transurban-submission-inquiry-qld.pdf
[35] Transurban. 2022. *Travel on our roads.* Retrieved 2022 from https://insights.transurban.com/travel/travel-on-our-roads/#toll-spend-data
[36] Hal Varian, Fredrik Wallenberg, and Glenn Woroch. 2005. The demographics of the do-not-call list [security of data]. *IEEE Security & Privacy* 3, 1 (2005), 34–39.
[37] Xiaokui Xiao, Gabriel Bender, Michael Hay, and Johannes Gehrke. 2011. iReduct: Differential privacy with reduced relative errors. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 229–240.
[38] Yonghui Xiao and Li Xiong. 2015. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1298–1309.
[39] Xinyu Yang, Teng Wang, Xuebin Ren, and Wei Yu. 2017. Survey on improving data utility in differentially private sequential data publishing. *IEEE Transactions on Big Data* 7, 4 (2017), 729–749.

Notation	Description
N	noise (added to wallet balances)
p	toll price
P	set of toll prices
pr	out-of-bounds probability
RE	relative error threshold / relative noise bound
SR	success rate
w	wallet balance
W	set of wallet balances
w_{min}/w_{max}	minimum/maximum possible wallet balance
w_o	obfuscated wallet balance
W_o	set of obfuscated wallet balances
W_p	set of plausible wallet balances
T_p	set of plausible traces
z	Absolute noise bound
Δ	sensitivity
λ	Laplace's scale

Table 5: Overview of Variables

[40] Keyu Zhu, Pascal Van Hentenryck, and Ferdinando Fioretto. 2021. Bias and variance of post-processing in differential privacy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11177–11184.

A NOTATION

We give an overview over our notation in Table 5.

B BACKGROUND DETAILS

B.1 Proof that the Laplace mechanism ensures ε-d_x-privacy

For $z \in Z$ and $x, x' \in X$, let pr_x and $pr_{x'}$ be the probability density function $\frac{1}{2\lambda} e^{\frac{-|x-z|}{\lambda}}$ and $\frac{1}{2\lambda} e^{\frac{-|x'-z|}{\lambda}}$ respectively. Let $\lambda = \frac{1}{\varepsilon}$. Then, we have

$$\frac{pr_x(z)}{pr_{x'}(z)} = \frac{\frac{1}{2\lambda} e^{\frac{-|x-z|}{\lambda}}}{\frac{1}{2\lambda} e^{\frac{-|x'-z|}{\lambda}}} = \frac{e^{-\varepsilon \cdot |x-z|}}{e^{-\varepsilon \cdot |x'-z|}} = e^{\varepsilon \cdot (|x'-z| - |x-z|)}$$

Using the triangle inequality for the metric d_x and as $d_x(x, x') = d_x(x', x)$, we get

$$\frac{pr_x(z)}{pr_{x'}(z)} \leq e^{\varepsilon \cdot |x-x'|} \leq e^{\varepsilon \cdot d(x,x')}$$

which completes our proof.

B.2 Relation of Parameters

Figure 3 shows how different parameters are connected in Eq. (3). The figure illustrates that:

(1) To increase the probability (1 - pr) of keeping relative error (re) below a certain threshold RE (given fixed Δ and w_{min}), larger values of ε are required.
(2) Smaller relative errors (RE) correspond to larger values of ε when w_{min}, pr, and Δ are fixed.

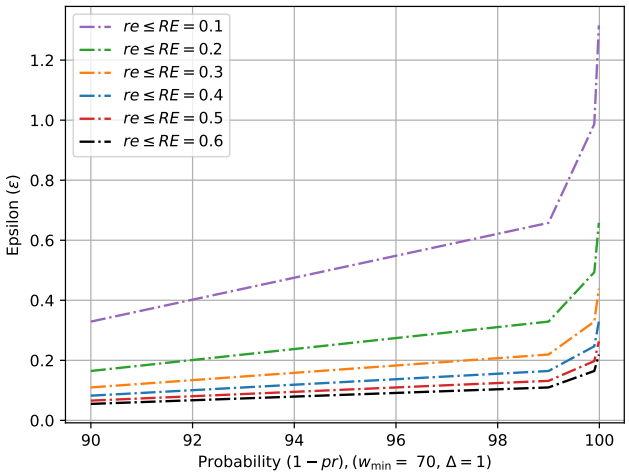


Figure 3: Relation of parameters in Eq. (3).

Algorithm 6 Obfuscation Algorithm based on d-Privacy

Input: $w, w_{min}, w_{max}, RE, pr$
Output: w_o

```
function METRIC_OBFUSCATION(w, w_min, w_max, RE, pr)
    λ ← −(RE · w_min)/ln(pr)
    N ← Lap(λ)
    w_o ← w + N
    if w_o < 0 then
        w_o ← 0
    else if w_o > w_max then
        w_o ← w_max
    else
        return w_o
    end if
    return w_o
end function
```

Note that the values shown in the graph are chosen arbitrarily. However, this does not impact the overall trends discussed in Section 2.2.

C DETAILS OF THE d-PRIVACY-BASED OBFUSCATION MECHANISM

C.1 The Obfuscation Algorithm

The details of the d-privacy based obfuscation algorithm are presented in Algorithm 6.

C.2 Impact of Parameters on Generated Noise

We now discuss how the parameters of the wallet obfuscation mechanism affect the generated noise, which defines the extra amount of cost users have to bear for privacy. We investigate the parameters out-of-bounds probability pr and minimum wallet balance w_{min}. In addition, we also use various fixed ε values, i.e., $\varepsilon \in \{0.1, 0.5, 1, 5\}$. With these parameters fixed, we evaluate how they influence the generated noise. Given a fixed pr, w_{min}, and ε, we also investigate

the impact of these parameters on RE and z using Eqs. (2) and (3). To keep the additional cost small, we cap the maximum amount of noise z relative to the minimum wallet balance, i.e.,

$$z \coloneqq RE \cdot w_{min} \tag{7}$$

For example, setting RE to 0.1 means that we set the maximum noise z to 10% of the minimum wallet balance w_{min}. Note that Eq. (7) only holds with probability $1 - pr$, i.e., out-of-bound noises are possible with a (very) small probability. This stems from the fact that to achieve differential privacy, one can't use a bounded noise function. This is also the reason why we employ post-processing in Algorithm 6. To analyze the distribution of generated noise, we generate 10^3 noises to examine the percentage of noises inside the threshold z. We deem this amount of noise sufficient, as we assume that a user will not participate in more than 1000 billing periods.

In our analysis we will subsequently vary either pr or w_{min}, while keeping the other value fixed. For that purpose we will use $pr = 10^{-3}$ and $w_{min} = 1$ as default values. Combined with different target privacy levels ε, we evaluate how the generated noise is affected. Note that the exact values of the examined parameters are not important, as they are only used to identify trends in the noise change when moving to larger/smaller parameter values. We show the results of our analysis in a table format, with the value under analysis (pr/w_{min}), the privacy level ε, the relative noise bound RE, the generated "non-outlier" noises, the generated "outlier" noises, and the absolute noise bound z as columns. Note that all noise values ("non-outliers", "outliers", and z) are in dollars. The gray and white colors (in a table) group rows that share the same values in the first column across all ε values to improve readability. The columns for non-outliers and outliers are determined using a box plot method that helps us to identify the range of generated noises. Non-outlier noise values fall within the range $(Q_1 - 1.5 \cdot IQR, Q_3 + 1.5 \cdot IQR)$, where Q_1 is the first quartile, Q_3 is the third quartile, and IQR is the interquartile range (the difference between the third and first quartiles). Outlier noise values N fall outside this range: For them, either (1) $N < Q_1 - 1.5 \cdot IQR$ or (2) $N > Q_3 + 1.5 \cdot IQR$ holds. In our analysis, we denote outliers as the tuple (N_{min}, N_{max}), where N_{min} and N_{max} are the minimum and maximum of outliers.

Parameter pr. The out-of-bounds probability pr denotes the probability that the generated noise falls outside the range $(-z, z)$. To ensure that the additional costs for users are not too high, pr should be small. But which values of pr are "small enough"? We evaluate the generated noise for $pr \in \{10^{-3}, 10^{-5}, 10^{-7}\}$. The results in Table 6 show that as pr decreases, the corresponding relative error RE and noise bound z increase. This means that to achieve the same privacy level (ε-DP) with a smaller pr, the generated noise is higher. This also means that the smaller pr, the higher the average additional cost to users. While this might suggest that the solution is to use a higher pr, i.e., $pr = 10^{-3}$, the situation is a bit more complicated. This is because one should also consider that the generated noise really falls in the range $(-z, z)$ to rule out the case that individual users have unrealistically high costs. But regarding this, the results show that for *smaller* pr the probability that the generated noise falls in the range $(-z, z)$ is higher. In particular, the table shows that for $pr = 10^{-7}$ and $pr = 10^{-5}$ the noises (including outliers) are within the range $(-z, z)$, while for $pr = 10^{-3}$ some outlier noises are outside of $(-z, z)$. Thus, the parameter pr should be chosen carefully to achieve a balance between a small additional cost and a high probability of staying within the desired noise range.

pr	ε	RE	non-outliers	outliers	z
0.001	0.1	69.1	(-28.0, 27.3)	(-66.5, 65.1)	69.1
0.001	0.5	13.8	(-5.6, 5.7)	(-12.2, 15.3)	13.8
0.001	1	6.9	(-2.7, 2.8)	(-4.7, 7.2)	6.9
0.001	5	1.4	(-0.6, 0.6)	(-1.7, 2.7)	1.4
1e-05	0.1	115.1	(-26.6, 27.4)	(-61.3, 61.8)	115.1
1e-05	0.5	23.0	(-5.3, 5.3)	(-12.4, 15.2)	23.0
1e-05	1	11.5	(-2.8, 2.8)	(-10.9, 7.0)	11.5
1e-05	5	2.3	(-0.6, 0.5)	(-1.2, 1.9)	2.3
1e-07	0.1	161.2	(-26.3, 24.5)	(-67.4, 70.2)	161.2
1e-07	0.5	32.2	(-6.0, 6.0)	(-12.8, 11.3)	32.2
1e-07	1	16.1	(-2.8, 2.9)	(-7.4, 8.2)	16.1
1e-07	5	3.2	(-0.5, 0.6)	(-1.3, 1.2)	3.2

Table 6: d-privacy: impact of pr on the generated noise

w_{min}	ε	RE	non-outliers	outliers	z
1	0.1	69.1	(-26.4, 25.1)	(-59.9, 66.2)	69.1
1	0.5	13.8	(-5.6, 5.6)	(-17.2, 11.7)	13.8
1	1	6.9	(-2.6, 2.7)	(-6.2, 5.8)	6.9
1	5	1.4	(-0.6, 0.6)	(-1.4, 1.2)	1.4
5	0.1	13.8	(-26.7, 26.3)	(-58.4, 85.4)	69.1
5	0.5	2.8	(-5.6, 5.6)	(-21.5, 10.0)	13.8
5	1	1.4	(-2.9, 2.9)	(-7.2, 5.9)	6.9
5	5	0.3	(-0.6, 0.6)	(-1.3, 1.6)	1.4
10	0.1	6.9	(-28.6, 28.4)	(-67.2, 64.1)	69.1
10	0.5	1.4	(-5.4, 5.6)	(-12.1, 11.2)	13.8
10	1	0.7	(-2.6, 2.6)	(-9.4, 6.8)	6.9
10	5	0.1	(-0.6, 0.6)	(-1.3, 1.2)	1.4

Table 7: d-privacy: impact of w_{min} on the generated noise

Parameter w_{min}. The minimum possible wallet balance w_{min} is directly defined by the ETC pricing scheme. For our analysis, we consider $w_{min} \in \{1, 5, 10\}$. Table 7 reveals that as w_{min} increases, the corresponding RE decreases for the same epsilon values. This indicates that the additional cost due to noise addition, relative to w_{min}, becomes smaller for the same level of privacy (ε-DP). Note that the actual noise bound z remains the same, because an increase in w_{min} is compensated by a decrease in RE (cp. Eq. (7)). This suggests that pricing schemes with a larger minimum possible wallet balance require less noise (relative to w_{min}), even though the actual noise bound remains the same for all w_{min}.

A Note on Outliers. As previously mentioned, the "outliers" and "non-outliers" columns in the tables are derived from the box plot method. Our results show that a significant portion of the generated noises, about 94%, comprises non-outliers. In contrast, only a small portion, i.e., approximately 6%, consists of outliers. This distribution occurs because the Laplace mechanism generates smaller noises with a much higher probability.

Summary of Results. While the out-of-bounds probability pr must be carefully chosen to achieve a reasonable privacy-cost trade-off, the privacy level ε plays the most critical role. Tables 6 and 7 all show that the higher ε is (corresponding to less privacy), the lower the noise bound z is. So TSPs should either set a desired privacy level ε and then see how much noise is needed, or set a desired noise bound z and then see how much privacy can be achieved. Whether or not there are acceptable combinations of ε and z that satisfy both a customer's need for privacy and a small additional cost depends on the pricing scheme and what additional costs users are willing to pay. As mentioned above, the question of what users consider an appropriate price for privacy is a separate line of research.

D DETAILS OF THE EXPONENTIAL OBFUSCATION MECHANISM

In this appendix, we explain the exponential obfuscation mechanism (Algorithm 1) in more detail, as well as Algorithms 7, 8, 9 and 10, which are sub-algorithms of Algorithm 1.

D.1 Algorithm EXPONENTIAL_OBFUSCATION

Algorithm 1 takes as its input ($trace \in T_p, T_p, \varepsilon, \alpha_{eucl} \in [0,1], \alpha_{sim} \in [0,1]$) and outputs the obfuscated trace obf_trace. We now explain the algorithm in detail.

Step 1: Precomputation. We first calculate the maximum possible Euclidean distance and the maximum possible similarity distance with COMPUTE_MAX_DIST (Algorithm 7) in line 2. Note that the precomputation step needs only to be executed once, as long as T_p stays the same.

Step 2: Score Calculation. Given an input trace $trace$, we now compute the score of the input trace and every possible output trace in lines 3 to 10, i.e., we calculate $u(trace, trace_j)$ for all $trace_j \in T_p$. We calculate $u(trace, trace_j)$ for a given $trace_j$ as follows: We first compute the euclidean distance d_{eucl} between the traces with COMPUTE_EUCLIDEAN (Algorithm 8). We also compute the similarity distance d_{sim} between the traces with COMPUTE_SIMILARITY (Algorithm 9). Afterwards we scale d_{eucl} to the range $[0,1]$ by dividing it by the maximum possible Euclidean distance (calculated in line 2). We also scale d_{sim} to $[0,1]$ analogously. Afterwards we compute the score $u(trace, trace_j)$ as

$$score \coloneqq (d'_{sim} \cdot \alpha_{sim} - d'_{eucl} \cdot \alpha_{eucl}),$$

where d'_{eucl} and d'_{sim} are the scaled versions of d_{eucl} and d_{sim}. Note that for a high score, the similarity distance should be high (which indicated very different traces), while the Euclidean distance should be small (which corresponds to small costs).

Step 3: Probability Calculation. Given the scores $u(trace, trace_j)$ for all $trace_j \in T_p$, we calculate for each $trace_j$ the probability that $trace_j$ gets selected as output trace in lines 11 an 12. According to the exponential mechanism, we calculate the the probability as specified in Eq. (5). We calculate the numerator of Eq. (5) in line 11 and then apply normalization in line 12.

Step 4: Selection of Output Trace. Given for each $trace_j \in T_p$ the probability that the input trace $trace$ gets mapped to $trace_j$ as output trace, we now just need to draw an output trace according to this probability distribution. Thus, we sample in line 13 an output trace obf_trace using the previously calculated probability distribution. Finally, we output obf_trace as obfuscated trace for $trace$.

Depending on whether the user sends a trace or a wallet balance to the TSP at the end of a billing period, he now may need to calculate the balance of the obfuscated trace afterwards.

Algorithm 7 Compute Maximum Distances

Input: T_p
Output: max_eucl, max_sim
1: **function** COMPUTE_MAX_DIST(T_p)
2: $max\_eucl \leftarrow 0$
3: $max\_sim \leftarrow 0$
4: **for all** $trace_i \in T_p$ **do**
5: **for all** $trace_j \in T_p$ **do**
6: $d_{eucl} \leftarrow$ COMPUTE_EUCLIDEAN($trace_i, trace_j$)
7: $d_{sim} \leftarrow$ COMPUTE_SIMILARITY($trace_i, trace_j$)
8: **if** $d_{eucl} > max\_eucl$ **then**
9: $max\_eucl \leftarrow d_{eucl}$
10: **end if**
11: **if** $d_{sim} > max\_sim$ **then**
12: $max\_sim \leftarrow d_{sim}$
13: **end if**
14: **end for**
15: **end for**
16: **return** max_eucl, max_sim
17: **end function**

Algorithm 8 Compute Euclidean Distance

Input: $trace_1 \in T_p, trace_2 \in T_p$
Output: $euclidean\_dist$
1: **function** COMPUTE_EUCLIDEAN($trace_1, trace_2$)
2: $w_1 \leftarrow$ GET_BALANCE($trace_1$)
3: $w_2 \leftarrow$ GET_BALANCE($trace_2$)
4: $euclidean\_dist \leftarrow |w_1 - w_2|$
5: **return** $euclidean\_dist$
6: **end function**

D.2 Algorithm COMPUTE_MAX_DIST

In Algorithm 7 we calculate the maximum possible Euclidean distance and the maximum possible similarity distance for a given set of traces T_p. To achieve that, we simply iterate over all possible pairs of traces, calculate the Euclidean distance and the sensitivity distance and check whether they are larger than are currently maximum. The algorithm outputs the maximum possible Euclidean distance as max_eucl and the maximum possible similarity distance as max_sim.

D.3 Algorithm COMPUTE_EUCLIDEAN

Algorithm 8 takes as input two traces, $trace_1$ and $trace_2$, and outputs the Euclidean distance between them. To achieve this, the algorithm uses the function GET_BALANCE to retrieve the corresponding

Algorithm 9 Compute Similarity Distance

Input: $trace_1 \in T_p, trace_2 \in T_p$
Output: sim_dist

function COMPUTE_SIMILARITY($trace_1, trace_2$)
  $sim\_dist \leftarrow 0$
  for all $(s_j, f_j) \in trace_1$ and $(s_j, f'_j) \in trace_2$ **do**
    if $f_j \neq 0$ and $f'_j \neq 0$ **then**
      $sim\_dist \leftarrow sim\_dist + (f_j - f'_j)^2$
    else if $f_j \neq 0$ and $f'_j == 0$ **then**
      $sim\_dist \leftarrow sim\_dist + (f_j)^2 + penalty^2$
    else if $f_j == 0$ and $f'_j \neq 0$ **then**
      $sim\_dist \leftarrow sim\_dist + (f'_j)^2 + penalty^2$
    end if
  end for
  $sim\_dist \leftarrow \sqrt{sim\_dist}$
  return sim_dist
end function

Algorithm 10 Compute Probabilities

Input: $arr\_score, \varepsilon, \Delta$
Output: arr_prob

function COMPUTE_PROB($arr\_score, \varepsilon, \Delta$)
  Declare $arr\_prob[|arr\_score|]$
  for all $score_i \in arr\_score$ **do**
    $arr\_prob[i] \leftarrow e^{(\varepsilon \cdot score_i)/(2 \cdot \Delta)}$
  end for
  return arr_prob
end function

wallet balances, denoted as w_1 and w_2. It then computes the Euclidean distance as $|w_1 - w_2|$ and stores the result in $euclidean\_dist$.

D.4 Algorithm COMPUTE_SIMILARITY

Algorithm 9 takes two traces, $trace_1$ and $trace_2$, as input and computes the similarity distance between them, denoted as sim_dist. To calculate this distance, the algorithm compares both the toll stations visited and their respective frequencies in the two traces. We denote a trace as

$$trace = \{(s_1, f_1), (s_2, f_2), \ldots, (s_l, f_l)\},$$

where each tuple (s_i, f_i) represents a toll station s_i and its corresponding frequency f_i. A frequency $f_i = 0$ in the tuple $(s_i, f_i) \in trace$ indicates that the toll station s_i has not been visited at all. In general, the Similarity distance between two points $P = (x_1, y_1, z_1, \ldots)$ and $Q = (x_2, y_2, z_2, \ldots)$ in an n-dimensional space is given by:

$$d(P, Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 + \ldots}$$

For each tuple $(s_j, f_j) \in trace_1$ and $(s_j, f'_j) \in trace_2$, the algorithm checks if both f_j and f'_j are non-zero. If so, it computes the distance as $sim\_dist = sim\_dist + (f_j - f'_j)^2$. Note that sim_dist is initialized to zero.

If the frequency of one trace is zero, while it non-zero in the other trace, an additional "penalty" is added. This reflects the intuition that traces that differ in terms of the toll stations visited cause more uncertainty than those that differ only in frequency values. Note that the penalty value can also be set to zero.

Finally, the algorithm takes the root of the final distance and returns the result as the output.

Remark 5 (On choosing the penalty value). Choosing an appropriate penalty value depends on what one assumes the background knowledge of the attacker to be. We differentiate two cases:

(1) We assume that the attacker knows which toll stations a user visited during the billing period, for example because he knows the home and work address of the user. Therefore, we only want to hide how often the user has passed the toll stations.
(2) We assume that the attacker does not possess such background knowledge and thus also want to hide which toll stations where visited during the billing period.

If (1) is the case, the penalty value should set very high. In that case, the obfuscation mechanism likely selects an output trace that visits the same toll station as the input trace, just with different frequencies. If (2) is the case, the penalty should be set to 0. In that case, the obfuscation algorithm ignores whether the output trace has the same visited toll stations as the input trace.

Note that we use $penalty = 0$ since we assume that the attacker does not have this kind of background knowledge.

D.5 Algorithm COMPUTE_PROB

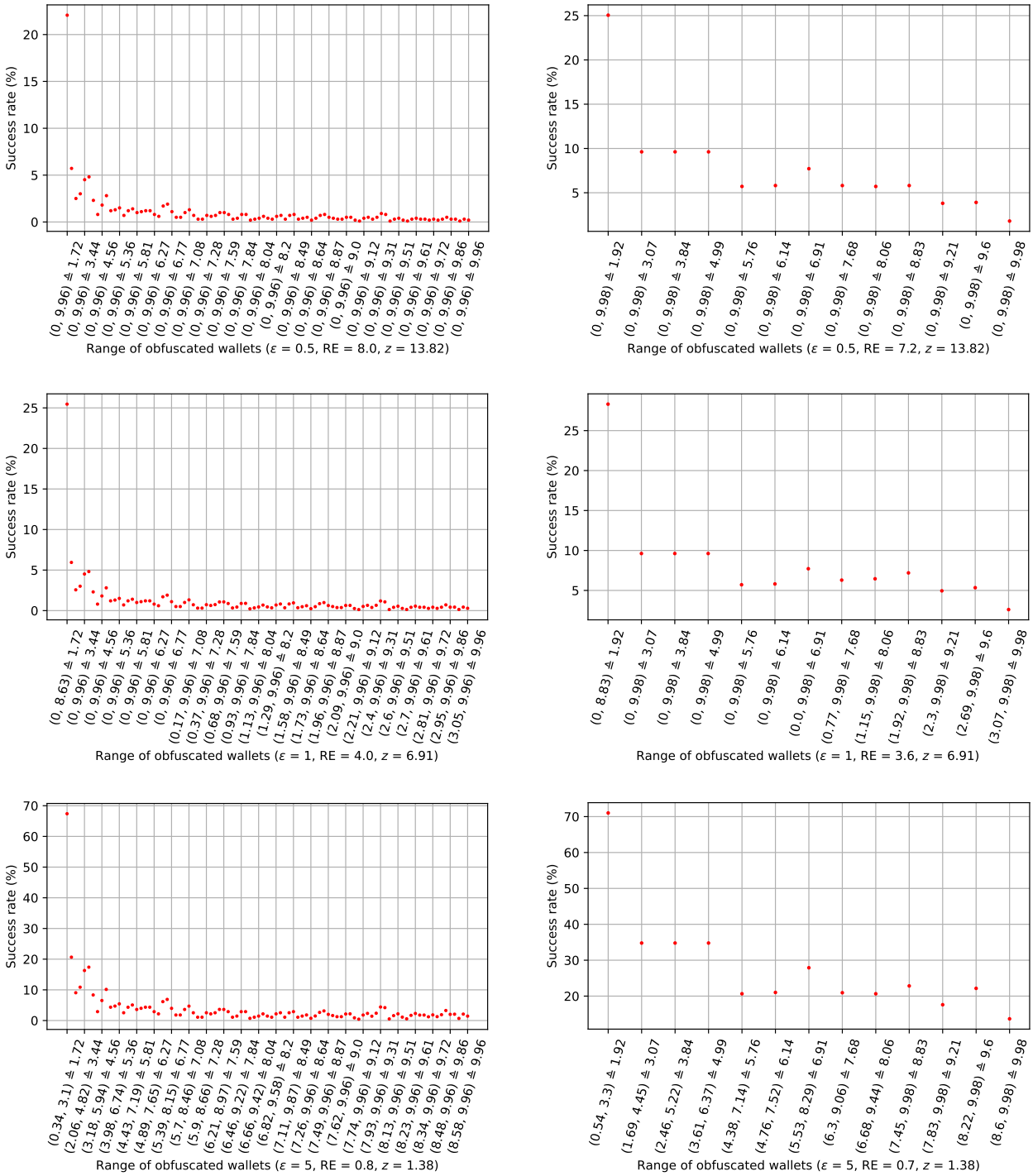
Algorithm 10 takes as input the list of scores arr_score, the parameter ε, and the sensitivity Δ, and it outputs a list arr_prob of probabilities. For each $score_i$ in arr_score, the algorithm computes the corresponding probability as $e^{(\varepsilon \cdot score_i)/(2 \cdot \Delta)}$ (according to the exponential mechanism, cp. the numerator in Eq. (5)) and stores it in the list arr_prob. Finally, the algorithm returns arr_prob as output.

E FURTHER EVALUATION OF d-PRIVACY

In this appendix, we present the results for the *wallet recovery attack* alone (Step 2 of our evaluation of d-privacy in Section 5.1).

Brisbane. In Fig. 4a the results for Brisbane are depicted. Overall, they are pretty similar to the results of the complete deobfuscation attack (cp. Fig. 1a). When taking a closer look, the success rates in the complete deobfuscation attack are a bit lower (or equal) than for the wallet recovery attack alone. This is because many wallet balances have different traces they could belong to (although some wallet balances only have on trace), so the adversary has to additionally figure out the correct trace.

Melbourne. For the wallet recovery attack on Melbourne, Fig. 4b shows a trend similar to Fig. 4a, but with a notable difference: the overall success rate is higher compared to the Brisbane case study. For $\varepsilon \in \{0.5, 1\}$, the difference is very small because the success rates remains close to zero for all wallet balances as well. The only considerable difference is for $\varepsilon = 5$, where the success rate can be >10% even for high wallet balances. This is because the precomputed ranges of deobfuscated wallet balances have less *overlapping* in the Melbourne case study (cp. Section 4.1.3) and the small noise ($z = 1.38$) that is added for $\varepsilon = 5$ is not enough to span many wallet balances.



(a) Brisbane **(b) Melbourne**

Figure 4: d-privacy: Each graph shows the success rate of the *wallet recovery attack* w.r.t a certain ε and corresponding RE, z. On the x-axis, each range (in dollars) includes all possible obfuscated wallets (w_o), linked with a certain original wallet (w). For example, the first x-axis value on the lower left graph (Brisbane with $\varepsilon = 5$) depicts the wallet with balance $1.72. For this wallet, the obfuscated wallet balances fall into the range $[\$0.34, \$3.1]$. On the y-axis the success rate of the attack is shown, i.e., the percentage of successful wallet recoveries. (Brisbane and Melbourne case study)

F FURTHER EVALUATION OF THE EXPONENTIAL MECHANISM

Here we further evaluate the exponential mechanism using different weights for euclidean distance and similarity distance. While we looked at ($\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25$) in Section 5.2, we will now also consider ($\alpha_{eucl} = 1, \alpha_{sim} = 0$) and ($\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5$).

F.1 Parameters ($\alpha_{eucl} = 1$, $\alpha_{sim} = 0$)

The privacy analysis for Brisbane and Melbourne can be found in Fig. 5. The cost analysis for Brisbane can be found in Table 8 and the one for Melbourne in Table 9.

F.2 Parameters ($\alpha_{eucl} = 0.5$, $\alpha_{sim} = 0.5$)

The privacy analysis for Brisbane and Melbourne can be found in Fig. 6. The cost analysis for Brisbane can be found in Table 10 and the one for Melbourne in Table 11.

F.3 Privacy and Cost Analysis

When considering *privacy*, Figs. 5 and 6 do not differ much from Fig. 2, as all provide pretty low success rates.

When comparing the *cost* (Tables 3, 4, 8, 9, 10 and 11,), it can be seen that the higher α_{sim} is, the higher the average costs are. This is due to the fact that when using a higher α_{sim}, obfuscated traces are favored that differ from the original trace in terms of the toll stations visited, and therefore also have a different wallet balance than the original trace.

G SIMILAR TRACE ATTACK

Instead of evaluating the success of an adversary based on whether he can find the *exact* original trace, one could also consider a more relaxed attack, where the adversary aims to find a trace that is *similar* to the original trace. We call this a "similar trace attack".

In this section, we first provide some details on how this similar trace attack is executed, before providing the results. To be able to evaluate the results, we compare an adversary that uses the similar trace attack with an adversary that just guesses a random trace. For the comparison with the random attacker, we describe in Appendix G.1 how to obtain the *average similarity of traces*. In Appendix G.2 we then provide the algorithms for the similar trace attack. Finally, in Appendix G.3 we evaluate the similar trace attack.

G.1 Average Similarity Between Traces

In Algorithm 11, given a set of plausible traces T_p, we calculate the average similarity between the traces. For that, we iterate over all possible pairs of traces and compute their similarity difference with Algorithm 9. Afterwards we calculate the average *avg* over all the differences.

G.2 Executing the Similar Trace Attack

We describe the similar trace attack for both d-privacy and the exponential mechanism. Basically, both algorithms get an original trace as input, then obfuscate the trace with the corresponding obfuscation mechanism and afterwards try to deobfuscate it using the corresponding deobfuscation algorithm. Then the similarity distance between the original trace and the deobfuscated trace, i.e., the attacker's guess for the original trace, is calculated. This procedure is repeated *num* times to get a good estimate of the attacker's success.

Algorithm 11 Compute Average Similarity Between Traces

Input: T_p
Output: *avg*

function COMPUTE_AVERAGE_SIMILARITY(T_p)
  Initialize arr_d_{sim} as empty list
  for all $trace_i \in T_p$ **do**
    for all $trace_j \in T_p$ **do**
      $d_{sim} \leftarrow$ COMPUTE_SIMILARITY($trace_i, trace_j$)
      Append d_{sim} to arr_d_{sim}
    end for
  end for
  $avg \leftarrow$ AVERAGE(arr_d_{sim}) \\Compute average of list
  return *avg*
end function

Algorithm 12 Relaxed Attack on one Trace for d-Privacy

Input: $trace \in T_p, w_{min}, w_{max}, RE, pr, num \in \mathbb{N}, list\_ranges$
Output: arr_d_{sim}

function METRIC_RELAXED_ATK($trace, w_{min}, w_{max}, RE, pr, num, list\_ranges$)
  Initialize arr_d_{sim} as empty list
  $w \leftarrow$ GET_BALANCE($trace$)
  for i from 1 to *num* **do**
    $obf\_balance \leftarrow$ METRIC_OBFUSCATION($w, w_{min}, w_{max}, RE, pr$)
    $w_c \leftarrow$ METRIC_WALLET_RECOVERY_ATTACK($obf\_balance, list\_ranges$)
    $deobf\_trace \leftarrow$ TSD_ATTACK(w_c)
    $d_{sim} \leftarrow$ COMPUTE_SIMILARITY($trace, deobf\_trace$)
    Append d_{sim} to arr_d_{sim}
  end for
  return arr_d_{sim}
end function

The similar trace attack for d-privacy is depicted in Algorithm 12. The similar trace attack for the exponential mechanism is depicted in Algorithm 13.

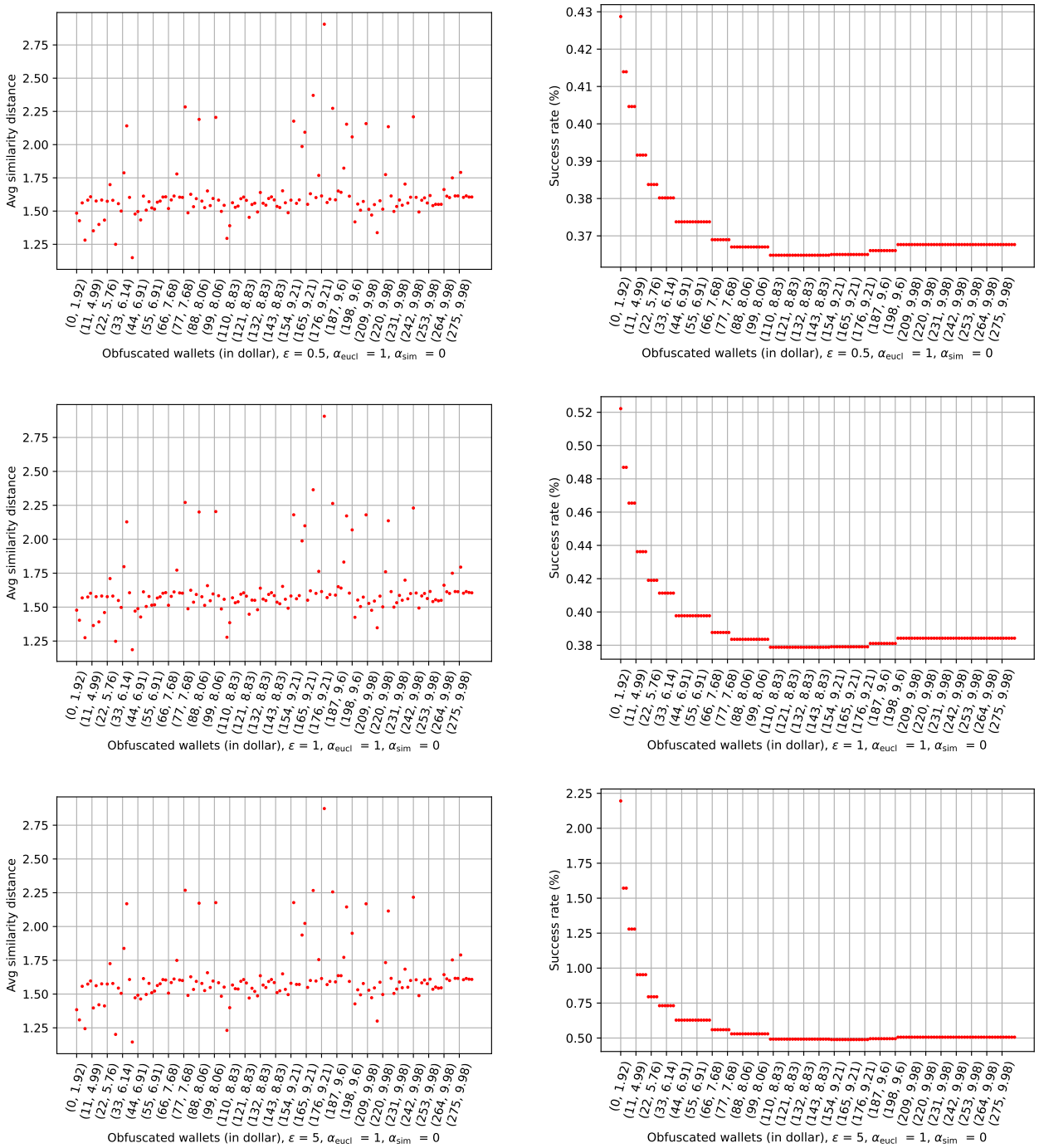
G.3 Evaluating the Similar Trace Attack

We describe the results of the similar trace attack for both d-privacy and the exponential mechanism.

G.3.1 *Evaluation Approach.*

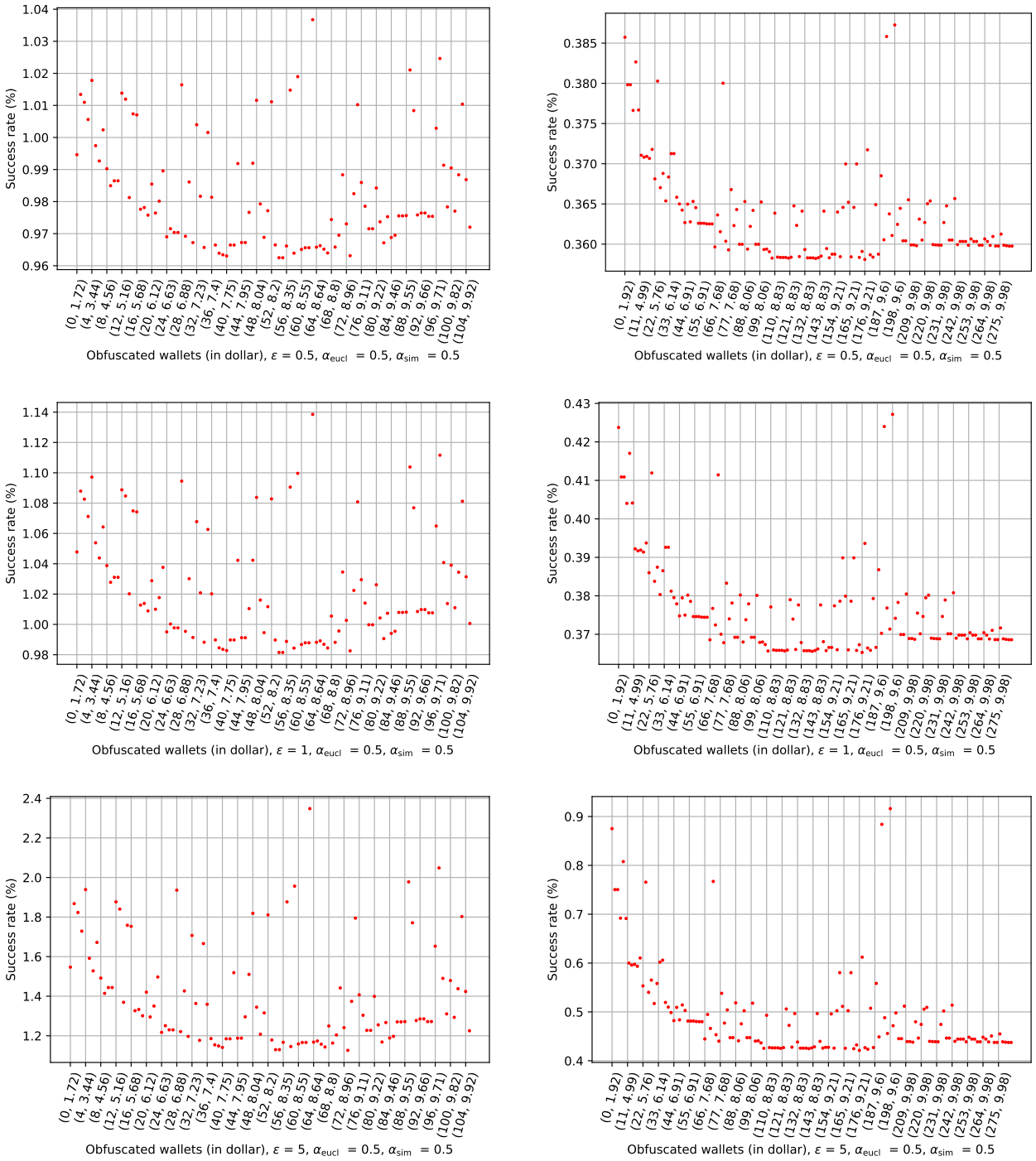
***d*-Privacy.** Our evaluation of the similar trace attack for d-privacy proceeds as follows:

(1) Fix parameters T_p, w_{min}, w_{max}, RE, pr, num.
(2) Precomputation: Execute Algorithm 2: $list\_ranges \leftarrow$ PRECOMPUTATION_METRIC_ATTACK(W_p, RE) to get the list of ranges $list\_ranges$
(3) For each $trace \in T_p$, we execute Algorithm 12 to get a list arr_d_{sim}. Algorithm 12 basically obfuscates the original trace $trace$, then deobfuscates it again and calculates the



(a) Brisbane **(b) Melbourne**

Figure 5: Exponential mechanism: Each graph shows the success rate of deobfuscation for $\varepsilon \in \{0.5, 1, 5\}$ and for $(\alpha_{eucl} = 1, \alpha_{sim} = 0)$. The tuples on the x-axis indicate $(id, wallet)$ where id distinguishes identical wallets with different traces. (Brisbane and Melbourne case study).



(a) Brisbane **(b) Melbourne**

Figure 6: Exponential mechanism: Each graph shows the success rate of deobfuscation for $\varepsilon \in \{0.5, 1, 5\}$ and for $(\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5)$. The tuples on the x-axis indicate $(id, wallet)$ where id distinguishes identical wallets with different traces. (Brisbane and Melbourne case study)

ε	(id, wallet)	non_outliers	outliers
0.5	(0, 1.72)	(1.5, 8.2)	(0.0, 1.1)
0.5	(10, 5.11)	(0.0, 4.9)	N
0.5	(20, 6.12)	(0.0, 4.4)	N
0.5	(30, 7.08)	(0.0, 4.4)	(5.4, 5.4)
0.5	(40, 7.75)	(0.0, 3.7)	(4.3, 6.0)
0.5	(50, 8.14)	(0.0, 3.6)	(3.7, 6.4)
0.5	(60, 8.55)	(0.0, 3.4)	(3.6, 6.8)
0.5	(70, 8.9)	(0.0, 4.4)	(4.5, 7.2)
0.5	(80, 9.22)	(0.0, 5.1)	(5.8, 7.5)
0.5	(90, 9.57)	(0.0, 6.4)	(6.7, 7.9)
0.5	(100, 9.82)	(0.0, 6.4)	(6.6, 8.1)

(a) $\varepsilon = 0.5$

ε	(id, wallet)	non_outliers	outliers
1	(0, 1.72)	(1.0, 8.2)	(0.0, 0.0)
1	(10, 5.11)	(0.0, 4.9)	N
1	(20, 6.12)	(0.0, 4.4)	N
1	(30, 7.08)	(0.0, 4.4)	(5.4, 5.4)
1	(40, 7.75)	(0.0, 3.7)	(4.3, 6.0)
1	(50, 8.14)	(0.0, 3.2)	(3.6, 6.4)
1	(60, 8.55)	(0.0, 2.9)	(3.0, 6.8)
1	(70, 8.9)	(0.0, 4.0)	(4.3, 7.2)
1	(80, 9.22)	(0.0, 4.7)	(4.8, 7.5)
1	(90, 9.57)	(0.0, 5.2)	(5.5, 7.9)
1	(100, 9.82)	(0.0, 6.6)	(7.0, 8.1)

(b) $\varepsilon = 1$

ε	(id, wallet)	non_outliers	outliers
5	(0, 1.72)	(0.0, 8.2)	N
5	(10, 5.11)	(0.0, 4.9)	N
5	(20, 6.12)	(0.0, 4.4)	N
5	(30, 7.08)	(0.0, 4.2)	(4.4, 5.4)
5	(40, 7.75)	(0.0, 3.7)	(4.3, 6.0)
5	(50, 8.14)	(0.0, 3.0)	(3.0, 5.5)
5	(60, 8.55)	(0.0, 2.7)	(2.9, 6.8)
5	(70, 8.9)	(0.0, 2.1)	(2.3, 7.2)
5	(80, 9.22)	(0.0, 2.6)	(2.8, 7.5)
5	(90, 9.57)	(0.0, 3.5)	(3.7, 7.9)
5	(100, 9.82)	(0.0, 4.1)	(4.3, 8.1)

(c) $\varepsilon = 5$

Table 8: Exponential mechanism for Brisbane case study: Each row shows the cost (in dollars) w.r.t a wallet (associated with a trace) and a fixed ε, using ($\alpha_{eucl} = 1, \alpha_{sim} = 0$). The id in $(id, wallet)$ distinguishes identical wallets with different traces.

ε	(id, wallet)	non_outliers	outliers
0.5	(0, 1.92)	(3.1, 8.1)	(0.0, 1.9)
0.5	(26, 5.76)	(0.0, 4.2)	N
0.5	(52, 6.91)	(0.0, 5.0)	N
0.5	(78, 7.68)	(0.0, 3.8)	(4.6, 5.8)
0.5	(104, 8.06)	(0.0, 3.1)	(4.2, 6.1)
0.5	(130, 8.83)	(0.0, 1.9)	(2.7, 6.9)
0.5	(156, 9.21)	(0.0, 3.1)	(3.5, 7.3)
0.5	(182, 9.6)	(0.0, 3.8)	(4.6, 7.7)
0.5	(208, 9.98)	(0.0, 5.0)	(6.1, 8.1)
0.5	(234, 9.98)	(0.0, 5.0)	(6.1, 8.1)
0.5	(260, 9.98)	(0.0, 4.2)	(5.0, 8.1)

(a) $\varepsilon = 0.5$

ε	(id, wallet)	non_outliers	outliers
1	(0, 1.92)	(1.1, 8.1)	(0.0, 0.0)
1	(26, 5.76)	(0.0, 4.2)	N
1	(52, 6.91)	(0.0, 5.0)	N
1	(78, 7.68)	(0.0, 3.8)	(4.6, 5.8)
1	(104, 8.06)	(0.0, 3.1)	(4.2, 6.1)
1	(130, 8.83)	(0.0, 1.9)	(2.7, 6.9)
1	(156, 9.21)	(0.0, 2.3)	(3.1, 7.3)
1	(182, 9.6)	(0.0, 2.7)	(3.5, 7.7)
1	(208, 9.98)	(0.0, 4.2)	(5.0, 8.1)
1	(234, 9.98)	(0.0, 4.2)	(5.0, 8.1)
1	(260, 9.98)	(0.0, 5.0)	(6.1, 8.1)

(b) $\varepsilon = 1$

ε	(id, wallet)	non_outliers	outliers
5	(0, 1.92)	(0.0, 8.1)	N
5	(26, 5.76)	(0.0, 4.2)	N
5	(52, 6.91)	(0.0, 5.0)	N
5	(78, 7.68)	(0.0, 3.8)	(4.6, 4.6)
5	(104, 8.06)	(0.0, 3.1)	(4.2, 5.0)
5	(130, 8.83)	(0.0, 1.9)	(2.7, 6.9)
5	(156, 9.21)	(0.0, 2.3)	(3.1, 7.3)
5	(182, 9.6)	(0.0, 0.8)	(1.5, 7.7)
5	(208, 9.98)	(0.0, 2.3)	(3.1, 6.9)
5	(234, 9.98)	(0.0, 2.3)	(3.1, 8.1)
5	(260, 9.98)	(0.0, 2.3)	(3.1, 8.1)

(c) $\varepsilon = 5$

Table 9: Exponential mechanism for Melbourne case study: Each row shows the cost (in dollars) w.r.t a wallet (associated with a trace) and a fixed ε, using ($\alpha_{eucl} = 1, \alpha_{sim} = 0$). The id in $(id, wallet)$ distinguishes identical wallets with different traces.

ε	(id, wallet)	non_outliers	outliers
0.5	(0, 1.72)	(1.5, 8.2)	(0.0, 1.1)
0.5	(10, 5.11)	(0.0, 4.9)	N
0.5	(20, 6.12)	(0.0, 4.4)	N
0.5	(30, 7.08)	(0.0, 4.4)	(5.4, 5.4)
0.5	(40, 7.75)	(0.0, 3.7)	(4.3, 6.0)
0.5	(50, 8.14)	(0.0, 3.6)	(3.6, 6.4)
0.5	(60, 8.55)	(0.0, 3.6)	(4.0, 6.8)
0.5	(70, 8.9)	(0.0, 4.8)	(5.5, 7.2)
0.5	(80, 9.22)	(0.0, 5.1)	(5.8, 7.5)
0.5	(90, 9.57)	(0.0, 6.1)	(6.4, 7.9)
0.5	(100, 9.82)	(0.0, 6.4)	(6.6, 8.1)

(a) $\varepsilon = 0.5$

ε	(id, wallet)	non_outliers	outliers
1	(0, 1.72)	(1.7, 8.2)	(0.0, 1.5)
1	(10, 5.11)	(0.0, 4.9)	N
1	(20, 6.12)	(0.0, 4.4)	N
1	(30, 7.08)	(0.0, 4.4)	(5.4, 5.4)
1	(40, 7.75)	(0.0, 3.7)	(4.3, 6.0)
1	(50, 8.14)	(0.0, 3.2)	(3.6, 6.4)
1	(60, 8.55)	(0.0, 3.4)	(3.6, 6.8)
1	(70, 8.9)	(0.0, 4.0)	(4.3, 7.2)
1	(80, 9.22)	(0.0, 4.8)	(5.1, 7.5)
1	(90, 9.57)	(0.0, 6.1)	(6.4, 7.9)
1	(100, 9.82)	(0.0, 5.7)	(6.4, 8.1)

(b) $\varepsilon = 1$

ε	(id, wallet)	non_outliers	outliers
5	(0, 1.72)	(0.0, 8.2)	N
5	(10, 5.11)	(0.0, 4.9)	N
5	(20, 6.12)	(0.0, 4.4)	N
5	(30, 7.08)	(0.0, 4.2)	(4.4, 5.4)
5	(40, 7.75)	(0.0, 3.7)	(4.3, 6.0)
5	(50, 8.14)	(0.0, 3.0)	(3.2, 6.4)
5	(60, 8.55)	(0.0, 2.7)	(2.9, 6.8)
5	(70, 8.9)	(0.0, 2.9)	(3.0, 7.2)
5	(80, 9.22)	(0.0, 3.5)	(3.7, 7.5)
5	(90, 9.57)	(0.0, 4.7)	(5.0, 7.9)
5	(100, 9.82)	(0.0, 5.4)	(6.4, 8.1)

(c) $\varepsilon = 5$

Table 10: Exponential mechanism for Brisbane case study: Each row shows the cost (in dollars) w.r.t a wallet (associated with a trace) and a fixed ε, using ($\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5$). The id in $(id, wallet)$ distinguishes identical wallets with different traces.

similarity between the deobfuscated trace and the original trace. This is repeated *num* times and the similarity after each deobfuscation is stored in an element of arr_d_{sim}.

ε	(id, wallet)	non_outliers	outliers
0.5	(0, 1.92)	(1.1, 8.1)	(0.0, 0.0)
0.5	(26, 5.76)	(0.0, 4.2)	N
0.5	(52, 6.91)	(0.0, 5.0)	N
0.5	(78, 7.68)	(0.0, 3.8)	(4.6, 5.8)
0.5	(104, 8.06)	(0.0, 3.1)	(4.2, 6.1)
0.5	(130, 8.83)	(0.0, 1.9)	(2.7, 6.9)
0.5	(156, 9.21)	(0.0, 3.1)	(3.5, 7.3)
0.5	(182, 9.6)	(0.0, 3.8)	(4.6, 7.7)
0.5	(208, 9.98)	(0.0, 5.0)	(6.1, 8.1)
0.5	(234, 9.98)	(0.0, 5.0)	(6.1, 8.1)
0.5	(260, 9.98)	(0.0, 5.0)	(6.1, 8.1)

(a) $\varepsilon = 0.5$

ε	(id, wallet)	non_outliers	outliers
1	(0, 1.92)	(3.1, 8.1)	(0.0, 1.9)
1	(26, 5.76)	(0.0, 4.2)	N
1	(52, 6.91)	(0.0, 5.0)	N
1	(78, 7.68)	(0.0, 3.8)	(4.6, 5.8)
1	(104, 8.06)	(0.0, 3.1)	(4.2, 6.1)
1	(130, 8.83)	(0.4, 1.2)	(0.0, 6.9)
1	(156, 9.21)	(0.0, 2.3)	(3.1, 7.3)
1	(182, 9.6)	(0.0, 3.8)	(4.6, 7.7)
1	(208, 9.98)	(0.0, 4.2)	(5.0, 8.1)
1	(234, 9.98)	(0.0, 5.0)	(6.1, 8.1)
1	(260, 9.98)	(0.0, 5.0)	(6.1, 8.1)

(b) $\varepsilon = 1$

ε	(id, wallet)	non_outliers	outliers
5	(0, 1.92)	(1.1, 8.1)	(0.0, 0.0)
5	(26, 5.76)	(0.0, 4.2)	N
5	(52, 6.91)	(0.0, 5.0)	N
5	(78, 7.68)	(0.0, 3.8)	(4.6, 5.8)
5	(104, 8.06)	(0.0, 3.1)	(4.2, 6.1)
5	(130, 8.83)	(0.0, 1.9)	(2.7, 6.9)
5	(156, 9.21)	(0.0, 2.3)	(3.1, 7.3)
5	(182, 9.6)	(0.0, 2.7)	(3.5, 7.7)
5	(208, 9.98)	(0.0, 4.2)	(5.0, 8.1)
5	(234, 9.98)	(0.0, 4.2)	(5.0, 8.1)
5	(260, 9.98)	(0.0, 4.2)	(5.0, 8.1)

(c) $\varepsilon = 5$

Table 11: Exponential mechanism for Melbourne case study: Each row shows the cost (in dollars) w.r.t a wallet (associated with a trace) and a fixed ε, using ($\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5$). The id in $(id, wallet)$ distinguishes identical wallets with different traces.

Algorithm 13 Relaxed Attack on one Trace for Exponential

Input: $trace \in T_p, T_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}, num \in \mathbb{N}, table$
Output: arr_d_{sim}

function EXP_RELAXED_ATK($trace, T_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}, num, table$)
  Initialize arr_d_{sim} as empty list
  for i from 1 to num **do**
    $obf\_trace \leftarrow$ EXPONENTIAL_OBFUSCATION($trace, T_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}$)
    $deobf\_trace \leftarrow$ EXP_ATTACK($obf\_trace, T_p, table$)
    $d_{sim} \leftarrow$ COMPUTE_SIMILARITY($trace, deobf\_trace$)
    Append d_{sim} to arr_d_{sim}
  end for
  return arr_d_{sim}
end function

(4) The results are plotted in a graph, where each element on the x-axis equals an original trace and the values of the corresponding arr_d_{sim} are depicted as a box plot on the y-axis.

We then compare the results of the similar trace attack with the average similarity between traces, i.e., an attacker that just randomly guesses a trace.

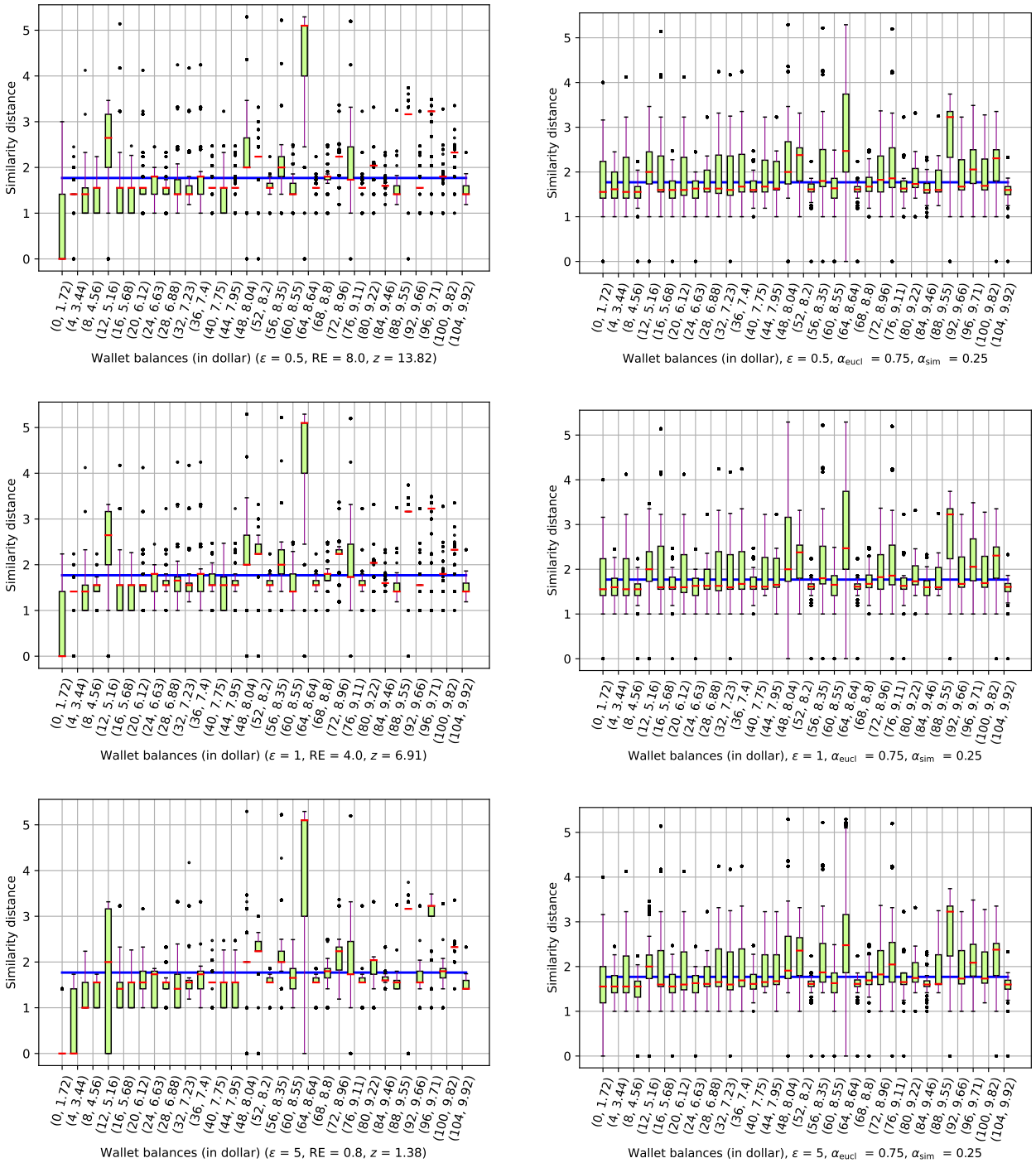
Exponential Mechanism. Our evaluation of the similar trace attack for the exponential mechanism proceeds as follows:

(1) Fix parameters T_p, ε, α_{eucl}, α_{sim}, num.
(2) Precomputation: Execute Algorithm 4 as $table \leftarrow$ PRECOMPUTATION_EXP_ATTACK($T_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}$) to get the table $table$ with all probabilities
(3) For each $trace \in T_p$, execute Algorithm 13($trace, T_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}, num, table$) to get a list arr_d_{sim}. Algorithm 13 basically obfuscates the original trace $trace$, then deobfuscates it again and calculates the similarity between the deobfuscated trace and the original same. This is repeated num times and the similarity after each deobfuscation is stored in an element of arr_d_{sim}.
(4) The results are plotted in a graph, where each element on the x-axis equals an original trace and the values of the corresponding arr_d_{sim} are depicted as a box plot on the y-axis.

We then compare the results oft the similar trace attack with the average similarity between traces, i.e., an attacker that just randomly guesses a trace.

G.3.2 ***Results.*** In Fig. 7 the results of $num = 1000$ iterations of the similar trace attack are depicted, in Fig. 7a for d-privacy and in Fig. 7b for the exponential mechanism. For both mechanism it can be seen that this attack does not fare much better than just guessing a random trace. Since the red line inside each box plot that depicts the median is in most cases close to the average similarity between traces (depicted as blue line), the chance for an adversary to recover a trace that is "similar" to the original trace is not much better than just guessing the original trace. But more medians (for both mechanism) are below the blue line, so the similar trace attack seems to yield better results than just guessing — even if only a little.

When comparing Fig. 7a and Fig. 7b in a bit more detail, it can be seen that for d-privacy the achieved similarity seems to fluctuate more and thus depends more on the input wallet, while for the exponential mechanism most input wallets yield a similar similarity. This is consistent with the results from the "exact trace attack", which we considered in Section 5.



(a) d-Privacy **(b) Exponential Mechanism with $(\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25)$**

Figure 7: Similar Trace Attack on the Brisbane Case Study: Each value on the x-axis depicts an original wallet, while the y-axis depicts box plots of the similarity between the trace the attacker outputs and the original trace. For a comparison, the average similarity between traces (obtained from Algorithm 11) is included as a blue line.