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

AMIRHOSSEIN ADAVOUDI JOLFAEI, Digital Learning Hub Luxembourg, Luxembourg
THOMAS ENGEL, University of Luxembourg, Luxembourg
VALERIE FETZER, Karlsruhe Institute of Technology, Germany and KASTEL SRL, Germany
ANDY RUPP, University of Luxembourg, Luxembourg and KASTEL SRL, Germany
STEFAN SCHIFFNER, Bonn-Rhein-Sieg University of Applied Sciences, Germany

Electronic toll collection (ETC) systems are becoming increasingly popular, but are inherently privacy-sensitive as they deal with users' location data. While prior research has proposed privacy-preserving ETC (PPETC) systems, which hide the individual toll fees from the toll service provider and provide it only with a total monthly fee, we study in this paper the actual privacy properties of PPETC schemes. Since prior work has shown that PPETC schemes may be insufficient to protect user privacy in real-world scenarios, we analyze the effectiveness of using an additional protection mechanism: applying a differential privacy (DP) mechanism that obscures the actual monthly toll fee by adding a small amount of noise. While this seems like a straightforward solution, it presents challenges. Adding noise to monthly fees can increase users' monetary costs, so the noise must be kept small. But since adding more noise intuitively means more privacy when applying DP, 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 DP mechanisms for categorical data, namely k-ary randomized response and the exponential mechanism, to evaluate their effectiveness in protecting users' toll station visits and determine the associated privacy costs. To investigate how well these mechanisms hide the visited toll stations, we design attacks on each protection mechanism that attempts to recover the toll station visits from an obscured monthly toll fee and evaluate its effectiveness in two real-world scenarios.

Additional Key Words and Phrases: Electronic toll collection, differential privacy, local DP, randomized response, exponential mechanism

1 Introduction

Electronic toll collection (ETC) is a technology primarily used to finance road infrastructure, but it can also support advanced functions such as congestion management and pollution reduction through dynamic pricing. ETC systems are implemented by *toll 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 appear more convenient for users than pre-payment systems. In post-payment ETC systems, the TSP needs to store certain sensitive information in order to charge users, including 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. This inherently allows the TSP to track the movements of each user, which has long been recognized as a privacy issue in the research community [26].

To address this problem, several *privacy-preserving ETC (PPETC)* schemes have been developed [4, 16, 19, 22, 27] that aim to 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, 9, 11]. This is because some information, such as the monthly toll

Authors' Contact Information: Amirhossein Adavoudi Jolfaei, Digital Learning Hub Luxembourg, Luxembourg, amir.adavoudi@trainer.dlh.lu; Thomas Engel, University of Luxembourg, Luxembourg, thomas.engel@uni.lu; Valerie Fetzer, Karlsruhe Institute of Technology, Germany and KASTEL SRL, Germany, valerie.fetzer@kit.edu; Andy Rupp, University of Luxembourg, Luxembourg and KASTEL SRL, Germany, andy.rupp@uni.lu; Stefan Schiffner, Bonn-Rhein-Sieg University of Applied Sciences, Germany, stefan.schiffner@h-brs.de.

fee, still needs to be disclosed for billing purposes. It has been shown in [2] that the monthly toll fee, when combined with publicly available background data such as road maps and usage statistics, can in some cases be sufficient to violate user privacy. More specifically, an attack on the real ETC system deployed in Brisbane reveals the toll stations visited by users with a monthly toll fee of ten or less Australian dollars (AUD) with a success rate of 94%. This attack is based on the observation that, given the monthly billing fee, reconstructing the visited toll stations is equivalent to solving the well-known *subset sum problem (SSP)*. While 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 [11], 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 [21]. In this work, we address this gap by investigating whether user privacy can be protected while keeping the associated cost relatively low. To this end, we examine two DP-based protection mechanisms and evaluate their performance in two real-world ETC infrastructures. We assume that users apply a *local* DP mechanism themselves before submitting their monthly fee to the TSP, such that the TSP only receives the obfuscated fee.[1]

Trade-Off between Privacy & Utility. When applying the *local differential privacy (LDP)* framework, one must carefully balance user privacy against data utility. It is straightforward to see that increasing the amount of noise improves privacy but reduces utility, and vice versa. In ETC systems, adding noise can also result in a higher billing fee for the user.[2] How much users are willing to pay for privacy is an different and open research question. While the monetary value of privacy has been empirically studied in certain domains, e.g., online privacy [18], location data privacy [3, 10, 35], or removal from marketers' call lists [33], [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 cost users are willing to bear for privacy. Instead, we focus on designing a mechanism that hides the exact monthly toll fee from the TSP and evaluate how much noise must be added to achieve a given level of privacy, expressed as ε-local differential privacy.

Our Contribution. We investigate two local differential privacy mechanisms designed for categorical data: *k-ary randomized response* [20] and the *exponential mechanism* [25]. Given that wallet balances in our setting can be represented as integer values within a known finite range, LDP mechanisms tailored to categorical data are a natural fit. Although we also evaluate the classical Laplace mechanism, which is more suitable for continuous data, our results (see Appendix D) indicate that k-ary randomized response and the exponential mechanism are better aligned with the characteristics of our scenario.

To assess the effectiveness of these LDP mechanisms in protecting users' toll station visit patterns, we simulate adversarial attacks targeting obfuscated monthly toll fees and quantify the adversary's success probability. Importantly, the adversary in our model operates solely on publicly available or easily accessible information, such as toll prices and obfuscated wallet data. This design choice ensures that the threat model remains broadly applicable and relevant for real-world deployments,

[1]We assume that the user also appends a Zero-Knowledge proof that they applied the LDP mechanism correctly. Details on this can be found in Section 3.3.
[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.

independent of the specific technology or operator behind the system. Consequently, any PPETC system should be designed to defend against such an adversary at a minimum.

In addition to evaluating the adversary's success chance, we analyze the privacy-utility trade-off by quantifying the cost incurred by users to conceal their toll expenditures. Our empirical evaluation, based on two real-world case studies, demonstrates that both mechanisms significantly reduce the adversary's success rate, while introducing some additional costs for users.

Structure of this Paper. Section 2 provides the necessary background, introducing key concepts related to electronic toll collection systems and local differential privacy. In Section 3, we present two mechanisms for toll fee obfuscation: k-ary randomized response and the exponential mechanism. Section 4 outlines our adversarial model and details the attacks targeting both obfuscation strategies. In Section 5, we evaluate the effectiveness of these attacks. Section 6 discusses the evaluation results and highlights directions for future research. Section 7, reviews related literature, and Section 8 concludes the paper with a summary of our main findings.

2 Background

We introduce some of the terms and concepts used in our ETC scenario, as well as define the necessary concepts from differential privacy.

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] bal, i.e., the sum of all prices of the visited toll stations. Given bal, the following equation holds:

$$bal = p_1 \cdot f_1 + p_2 \cdot f_2 + \cdots + p_l \cdot f_l, \quad f_j \in \mathbb{N}_0 \tag{1}$$

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.

$$w_l < p_1 \cdot f_1 + p_2 \cdot f_2 + \cdots + p_l \cdot f_l < w_u \tag{2}$$

The set of all solutions derived from Inequality 2 is denoted as W_p, where each element

$$w := (id, bal, trace) \in W_p \tag{3}$$

consists of a wallet id id, a wallet balance bal, 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 2 can be discarded if they are not possible given the road network.

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

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: We define the *cost* of privacy as the difference between the balance of the original wallet and the balance of the obfuscated wallet. This represents the additional amount a customer needs to tolerate in order to protect their privacy [8, 29, 30].

Subset Sum Problem (SSP): The SSP is an NP-complete problem [23], 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 [12], 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.

Local Differential Privacy. In the *local differential privacy (LDP)* model, each user performs the privacy mechanism themselves on their own data before sending it to the central data collector. This removes the need for trust in the central data collector, which is the TSP in our scenario. Informally, LDP ensures that the output of the privacy mechanism reveals only little about the real input, even to the data collector itself. The privacy parameter ε quantifies the level of privacy loss. A smaller value of ε indicates less privacy loss [14].

Definition 2.1 (ε-Local Differential Privacy (ε-LDP) [28])**.** A randomized mechanism M satisfies ε-LDP if and only if for any pairs of input values $x, x' \in D$, and for any possible output $y \in R$, it holds that

$$\Pr[M(x) = y] \leq e^{\varepsilon} \cdot \Pr[M(x') = y]. \tag{4}$$

In our case, the dataset D will be the set of plausible wallets W_p.

(K-ary) Randomized Response. One of the most widely used mechanisms for achieving ε-LDP over categorical data is *k-ary randomized response (KRR)* [20], which generalizes Warner's original technique [34] to a domain of size k instead of 2. The core idea of KRR is that either the true value is reported or a random one, whereas the true value is reported with a higher probability than a random one (except for $\varepsilon = 0$, where every value has the same probability to be chosen). This also means that the range and domain of KRR are the same set, i.e., $R = D$ holds.

Given an input $x \in D$, with $|D| = k$, the KRR mechanism outputs the following:

$$M_{\text{KRR}}^{k,\varepsilon,D}(x) = \begin{cases} x, & \text{with probability } \dfrac{e^{\varepsilon}}{e^{\varepsilon} + k - 1} \\ z \neq x, & \text{with probability } \dfrac{1}{e^{\varepsilon} + k - 1} \text{ for each } z \neq x \end{cases} \tag{5}$$

The Exponential Mechanism. The exponential mechanism, instead of adding some noise to the input, selects an output from a set of possible outcomes. Although it was originally proposed for standard DP, it can also be applied to local DP. In this case, each user randomly selects their output based on a *locally executable score function.* The exponential mechanism biases its choice toward higher-scoring outcomes, ensuring that no single input influences the output distribution too much and thereby satisfying ε-LDP.

The exponential mechanism for a domain D and outcome set R uses a scoring function $u : D \times R \longrightarrow \mathbb{R}$ which maps $(x \in D, r \in R)$ pairs to a real-valued score. More precisely, the exponential mechanism $M_{\text{EXP}}^{\varepsilon,u,R}(x)$, on input $x \in D$, samples an element r from R according to the following probability distribution [25]:

$$\Pr_{M_{\text{EXP}}^{\varepsilon,u,R}}[r \mid x] = \frac{\exp\left(\frac{\varepsilon \cdot u(x,r)}{2\cdot\Delta_u}\right)}{\sum_{i\in R}\exp\left(\frac{\varepsilon\cdot u(x,i)}{2\cdot\Delta_u}\right)} \tag{6}$$

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

$$\Delta_u = \max_{r\in R}\max_{x,x'} |u(x,r) - u(x',r)| \tag{7}$$

where x and x' are "neighboring inputs" [25].

In our case, all possible inputs and outputs are wallets, thus $D = R = W_p$. Note that we assume all wallets to be "neighbors", thus aiming to hide the user's complete trace from the adversary.

Laplace Mechanism. When using the Laplace mechanism for local DP, each user adds random noise, which is drawn from the Laplace distribution, to their own value before sending it to the central data collector.

Definition 2.2 (Laplace Mechanism for LDP [28])**.** Given a data item v, the mechanism outputs

$$M(v) = v + \text{Lap}\left(\frac{\Delta f}{\varepsilon}\right)$$

where Δf is the sensitivity of the function f and $\text{Lap}(\cdot)$ represents the Laplace distribution.

3 Wallet Obfuscation Mechanisms

In post-payment ETC systems, users typically settle their toll fees at the end of each billing period by submitting their wallet balance, which reflects the total cost of all toll stations visited during that period. Prior work [2, 9] has shown that revealing the exact wallet balance can compromise user privacy, as it may allow adversaries to reconstruct a user's trace with significant probability when combined with auxiliary information.

To address this issue, we introduce two *wallet obfuscation mechanisms* based on LDP. Both mechanisms are executed locally by the users before submitting their wallet balance to the TSP, as they rely solely on the user's own data

We focus on two mechanisms designed for categorical data: the k-ary randomized response (KRR) and the exponential mechanism. The Laplace mechanism, which is more suited for continuous data, is discussed separately in Appendix D.

3.1 Obfuscation based on the KRR Mechanism

First, we use k-ary randomized response (KRR) to let the user either submit their real wallet balance or a random balance that is "close" to the real one. The formal algorithm is presented in Algorithm 1. Suppose the user's original wallet is w_i. Then, the algorithm first computes the set of neighbors of w_i (including w_i itself), which has size k:

$$N \longleftarrow \textsc{neighbors\_set}(w_i, W_p, k) = \{w_{i-\lfloor(k-1)/2\rfloor}, \ldots, w_i, \ldots, w_{i+\lceil(k-1)/2\rceil}\}$$

Then, the algorithm samples an element of N according to the KRR mechanism (cp. Section 2.2) and outputs the resulting wallet. Note that the higher ε is, the higher the probability that the obfuscated wallet equals the original wallet.

Algorithm 1 Obfuscation based on k-ary Randomized Response (KRR)

```
Input: $w_i \in W_p, W_p, k \in \{1, \dots, |W_p|\}, \varepsilon$
Output: $w_o$
1: function KRR_OBFUSCATION($w_i, W_p, k, \varepsilon$)
2:     $N \leftarrow$ NEIGHBORS_SET($w_i, W_p, k$)
3:     $w_o \leftarrow M_{\text{KRR}}^{k,\varepsilon,N}(w_i)$
4:     return $w_o$
5: end function
```

3.2 Obfuscation based on the Exponential Mechanism

Next, we construct an obfuscation mechanism based on the exponential mechanism $M_{\text{EXP}}^{\varepsilon,u,R}(\cdot)$ (cp. Section 2.2). The pseudocode of the mechanism is shown in Algorithm 2. The core idea is that, given a fixed wallet $w_i \in W_p$, $M_{\text{EXP}}^{\varepsilon,u,R}$ selects a wallet from W_p that somewhat maximizes the score while guaranteeing privacy. Our scoring function $u : W_p \times W_p \to \mathbb{R}$ 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's balances
(2) The *similarity distance* d_{sim} measures how similar the wallet's 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 $w_j \in W_p$ is selected by $M_{\text{EXP}}^{\varepsilon,u,R}$ as $prob \coloneqq \exp\left(\frac{\varepsilon \cdot score}{2 \cdot \Delta}\right)$, where Δ is the sensitivity of the scoring function (in our case, $\Delta \coloneqq 1$). After doing this for all $w_j \in W_p$, $M_{\text{EXP}}^{\varepsilon,u,R}$ samples an obfuscated wallet 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. The complexity of Algorithm 2 is linear in the number of wallets, i.e., $|W_p|$. Additional details on $M_{\text{EXP}}^{\varepsilon,u,R}$ are given in Appendix A.

3.3 Integrating the Obfuscation Mechanisms into PPETC schemes

We now discuss how the obfuscation mechanisms introduced above can be securely integrated into a PPETC scheme, such as the one proposed in [16]. A central challenge lies in the generation of randomness required by these mechanisms, as they are inherently randomized algorithms. A straightforward approach would be to let users generate the randomness themselves. While this aligns with the notion that users are responsible for their own privacy, it introduces the risk of manipulation. Specifically, users could bias the randomness to produce minimal toll values, thereby reducing their fees and causing revenue loss for the TSP.

To mitigate this risk, a more robust solution involves joint randomness generation by both the user and the TSP. Let the user sample a random value r_u and the TSP sample r_t. A straightforward implementation would be to then let both parties exchange their random values, allowing the user to compute the obfuscation output using the combined randomness. However, this would expose r_u to the TSP, enabling it to reverse-engineer the obfuscation process by simulating the mechanism with some plausible wallet balances. Since the obfuscation algorithm becomes deterministic when the randomness is fixed, this could compromise the user's privacy.

To prevent this, we propose the following steps:

Algorithm 2 Obfuscation Algorithm based on the Exponential Mechanism

```
Input: $w_i \in W_p, W_p, \varepsilon, \alpha_{eucl} \in [0,1], \alpha_{sim} \in [0,1]$
Output: $w_o$
1: function EXPONENTIAL_OBFUSCATION($w_i, W_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}$)
2:     Parse $(id, bal, trace) \leftarrow w_i$
3:     Extract $T_p$ from $W_p$
4:     $(max\_eucl, max\_sim) \leftarrow$ COMPUTE_MAX_DIST($T_p$)
5:     Declare $arr\_score[|W_p|]$
6:     for all $w_j \in W_p$ do
7:         Parse $(id_j, bal_j, trace_j) \leftarrow w_j$
8:         $d_{eucl} \leftarrow$ COMPUTE_EUCLIDEAN($trace, trace_j$)
9:         $d_{sim} \leftarrow$ COMPUTE_SIMILARITY($trace, trace_j$)
10:        $d'_{eucl} \leftarrow d_{eucl}/max\_eucl$
11:        $d'_{sim} \leftarrow d_{sim}/max\_sim$
12:        $arr\_score[j] \leftarrow (d'_{sim} \cdot \alpha_{sim} - d'_{eucl} \cdot \alpha_{eucl})$
13:    end for
14:    $arr\_prob \leftarrow$ COMPUTE_PROB($arr\_score, \varepsilon, \Delta := 1$)
15:    $arr\_norm\_prob \leftarrow$ NORMALIZE($arr\_prob$)
16:    $trace_k \leftarrow$ SELECT_RAND($arr\_norm\_prob, T_p$)
17:    $w_o \leftarrow w_k$
18:    return $w_o$
19: end function
```

(1) The user samples r_u and the TSP samples r_t.
(2) The user sends a *commitment*[4] to r_U to the TSP, i.e., $com \leftarrow \text{COM.Commit}(r_U)$.
(3) The TSP responds by sending r_t in plaintext to the user.[5]
(4) The user computes the joint randomness $r = r_u \oplus r_t$ and uses it to obfuscate their wallet balance w_i, resulting in w_o.
(5) The user computes a *Zero-Knowledge proof*[6] Π that shows that there exist (w_i, r_U) for that the following relations holds:
    - $r = r_u \oplus r_t$
    - $r_u = \text{COM.Open}(com)$
    - $w_o = \text{Algorithm}(w_i; r)$ for Algorithm being the obfuscation algorithm
(6) The TSP, who already knows (r_T, com), then receives (w_o, Π), verifies the proof, and is thereby convinced that the user used the correct joint randomness $r = r_U \oplus r_T$ as obfuscation randomness without learning r_u itself.

This protocol represents one possible way to implement the obfuscation process within a PPETC scheme. It ensures that users cannot manipulate the randomness while preserving their privacy from the service provider.

4 Deobfuscation Attacks

In this section, we design attacks against the LDP-based wallet obfuscation mechanisms from Section 3. Later in Section 5 we will then evaluate their effectiveness.

[4]A commitment scheme allows a user to commit to a message m and publish the result, called commitment com, in a way that m is hidden from others, but also the user cannot claim a different m afterwards when he opens com.
[5]Note that these are the first two messages of a *Blum coin toss* [6]. The third message, where the user opens to commitment to reveal r_U to the TSP is omitted since this would lead to the TSP learning the obfuscation randomness.
[6]A Zero-Knowledge proof allows a party, e.g., the user, to convince another party, e.g., the TSP, that a statement is true (e.g., that the obfuscation was performed correctly), without revealing any information beyond the validity of the statement itself.

For all attacks, the adversary gets an obfuscated wallet (and some other information) as input and outputs a guess for the original wallet, which we will call *deobfuscated wallet*. In Section 5 we will consider an attack attempt successful if the deobfuscated wallet equals the original wallet.

4.1 Threat Model

Although the privacy level in the DP framework is parameterized by ε, it does not reflect the absolute level of privacy for a user, i.e., what can really be inferred from a user's secret [13, 15, 29]. 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, denoted as w_o, for which it wants to identify the correct (deobfuscated) wallet. 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. (2). For evaluation purposes we also assume that the adversary knows the parameters used for obfuscation.[7] These are (ε, k) for KRR and $(\varepsilon, \Delta, \alpha_{eucl}, \alpha_{sim})$ for the exponential mechanism. In summary, the adversary's knowledge is represented as $\mathcal{K}_{\text{KRR}} = \{w_o, P, W_p, \varepsilon, k\}$ for the KRR mechanism and $\mathcal{K}_{\text{EXP}} = \{w_o, P, W_p, \varepsilon, \Delta, \alpha_{eucl}, \alpha_{sim}\}$ for the exponential mechanism.

It's worth noting that the adversary in our model could realistically be *anyone*, since it only relies on publicly available or easily accessible information, such as toll prices and obfuscated wallet data. This makes the threat model broadly applicable and relevant for real-world scenarios. As a result, any PPETC system should be designed to defend against such an adversary at the very least. Moreover, our analysis is independent of any specific technology, meaning the adversary model applies regardless of who operates the system or how it is implemented.

4.2 Deobfuscation Attack on the KRR Mechanism

For the attack on the KRR mechanism we have to distinguish between the cases $\varepsilon = 0$ and $\varepsilon > 0$.

For $\varepsilon = 0$, the obfuscation mechanism uniformly random samples an element of the neighbor set N, therefore the only possible deobfuscation strategy is to also compute N (which is possible for $\mathcal{A}$) and to randomly guess a deobfuscated wallet from that set.

For $\varepsilon > 0$, the probability that the KRR obfuscation mechanism sampled the *original* wallet as obfuscated wallet is higher than the probability for any other wallet in N. Therefore, $\mathcal{A}$'s best option is to just select the obfuscated wallet as deobfuscated wallet, which is a maximum likelihood strategy.

4.3 Deobfuscation Attack on the Exponential Mechanism

The attack on the exponential mechanism includes two steps: (1) precomputation and (2) deobfuscation. In the precomputation phase (cp. Algorithm 3), $\mathcal{A}$ creates a table containing the probabilities that w_i got mapped to w_j during obfuscation, for $i, j \in \{0, \ldots, |W_p| - 1\}$. For that, the exponential *ob*fuscation algorithm (Algorithm 2) is executed for each $w_i \in W_p$ to get the probability that this wallet gets mapped to w_j, for all $w_j \in W_p$. The results are stored in a table, where the cell (i, j) contains the likelihood that w_i is mapped to w_j.

In the deobfuscation phase (cp. Algorithm 4), given an obfuscated wallet w_o, $\mathcal{A}$ simply selects the cell in the jth column that contains the maximum of the jth column. This cell holds the id of

[7]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.

Algorithm 3 Precomputation for Attack on the Exponential Mechanism

```
Input: $W_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}$
Output: $table$
1: function PRECOMPUTATION_EXP_ATTACK($W_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}$)
2:     Extract $T_p$ from $W_p$
3:     Declare $table[|T_p|][|T_p|]$                                  \\Create 2-dimensional array
4:     for all $trace_i \in T_p$ do
5:         $arr\_norm\_prob \leftarrow$ COMPUTE_NORMALIZED_PROB($trace_i, T_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}$)
           \\This executes EXPONENTIAL_OBFUSCATION until $arr\_norm\_prob$ is computed (line 15)
6:         $table[i] \leftarrow arr\_norm\_prob$
7:     end for
8:     return $table$
9: end function
```

Algorithm 4 Attack on the Exponential Mechanism

```
Input: $w_o \in W_p, W_p, table$
Output: $w_d$
1: function EXP_ATTACK($w_o, W_p, table$)
2:     Declare $column[|W_p|]$                                          \\Create array
3:     Let $j$ be the index for which $w_o = W_p[j]$ holds
4:     for $i$ from 0 to $|W_p| - 1$ do                            \\get $j$th column of $table$
5:         $column[i] \leftarrow table[i][j]$
6:     end for
7:     $index \leftarrow arg\_max(column)$                          \\Find maximum probability
8:     $w_d \leftarrow W_p[index]$                   \\Select wallet that has maximum probability
9:     return $w_d$
10: end function
```

the wallet with the highest probability of being the original wallet. Note that attack uses again a maximum likelihood strategy.

5 Evaluation

We now evaluate the effectiveness of our deobfuscation attacks from Section 4 against our wallet obfuscation mechanisms from Section 3 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 applying the KRR mechanism or the exponential mechanism to this scheme helps in hiding the toll station visits. Note that we do not include the Laplace mechanism here, since our evaluation results have shown that the Laplace mechanism is not the top choice for any of our metrics. Therefore, the analysis of the Laplace mechanism is deferred to Appendix D.

The adversary's goal in this section is to recover the original wallet of the user, given an obfuscated wallet. Remember that a wallet is always uniquely linked to a trace, so recovering the original wallet of a user equals recovering the trace of the user.

In Appendix E, we consider a more relaxed attack where the adversary tries to find a wallet/trace that is just *similar* to the original wallet/trace instead of finding the *exact* wallet/trace. The results show that both mechanisms perform only slightly better than random guessing. The difference is small, and both KRR and the exponential mechanism behave quite similarly under this relaxed attack model.

ε	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 11$
0.1	35.59%	21.65%	15.55%	12.14%	9.95%
0.5	45.19%	29.19%	21.56%	17.09%	14.15%
1	57.61%	40.46%	31.18%	25.36%	21.37%
2	78.70%	64.88%	55.19%	48.02%	42.49%

Table 1. Success rates for deobfuscation of KRR mechanism.

Parameters for the Brisbane Case Study. To evaluate our attacks, we utilize the actual parameters of Brisbane's ETC system [7, 31, 32], which has also been examined in [2]. We use the following parameters for our evaluation:

Toll prices (P): The 9 toll prices (in AUD) are as follows [31]: $P = \{1.72, 2.68, 2.84, 3.19, 4.09, 4.55, 5.11, 5.11, 5.46\}$.

Plausible wallets (W_p): Based on the toll prices, we can compute the set of plausible wallets W_p within the range $[\$0, \$10]$[8] using Eq. (2). Note that $|W_p| = 106$ (cp. Appendix B).

Parameters for the Melbourne Case Study. As a second real-world example, we examine a PPETC system based on the Melbourne ETC system [24], which has the following parameters:

Toll prices (P): We assume the following 19 toll prices (in AUD): $P = \{1.92, 1.92, 3.07, 3.07, 3.07, 3.07, 3.84, 3.84, 4.99, 6.14, 6.14, 6.91, 6.91, 6.91, 8.06, 9.98, 9.98, 9.98, 10.75\}$.

Plausible wallets (W_p): As for the Brisbane case study, we obtain all plausible wallet balances within the range $[\$1, \$10]$ with Eq. (2). Note that $|W_p| = 285$ (cp. Appendix B).

5.1 Evaluation of the KRR Mechanism

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

5.1.1 ***Privacy Analysis.*** We evaluate the privacy level of an individual by calculating the success rate of the deobfuscation attack from Section 4.2, for different privacy levels ε and neighbor set size k. More precisely, we evaluate $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and $k \in \{3, 5, 7, 9, 11\}$.

Brisbane and Melbourne Case Studies. Since the success rate of the KRR deobfuscation attack depends only on the parameters ε and k and is independent of the actual wallets, the success rates for attacking the Brisbane and Melbourne case studies are the same. An adversary using the strategy described in Section 4.2 outputs the obfuscated wallet as the deobfuscated wallet, which is correct with probability $\frac{e^{\varepsilon}}{e^{\varepsilon}+k-1}$. The resulting success chances can be seen in Table 1. It can easily be seen that smaller ε and larger k yield smaller deobfuscation success rates. However, the individual success rates differ greatly, the lowest being 9.95% (for $\varepsilon = 0.1$, $k = 11$) and the highest being 78.70% (for $\varepsilon = 2$, $k = 3$).

5.1.2 ***Cost Analysis.*** We also evaluate the amount of additional noise introduced by obfuscation for the same (ε, k) 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 k, we add noise to each plausible wallet w_i to obtain its associated obfuscated wallet w_o, using Section 3.1.

Step 2: Having obtained the obfuscated wallet, we compute the cost as $|\textsc{get\_balance}(w_o) - \textsc{get\_balance}(w_i)|$.

[8] 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.

	Brisbane						Melbourne					
	KRR with k =					EXP with	KRR with k =					EXP with
ε	3	5	7	9	11	$\alpha_{eucl} = 0.75$	3	5	7	9	11	$\alpha_{eucl} = 0.75$
0.1	0.05	0.09	0.14	0.18	0.22	2.04	0.02	0.03	0.05	0.06	0.08	1.89
0.5	0.04	0.09	0.13	0.17	0.21	2.01	0.01	0.03	0.04	0.05	0.06	1.83
1	0.03	0.07	0.11	0.15	0.19	1.96	0.01	0.02	0.03	0.03	0.04	1.80
2	0.02	0.04	0.07	0.10	0.14	1.89	0.00	0.01	0.01	0.02	0.02	1.71

Table 2. Average cost of obfuscation in AUD for all mechanisms considered in the main body, for different ε.

We repeat this 1000 times to get a good estimate of the costs.

Brisbane Case Study. In Fig. 1, the incurred costs are shown for the Brisbane case study, for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and $k \in \{3, 5, 7, 9, 11\}$. The figure uses box plots to visualize the distribution of costs, where the yellow line inside each box indicates the *median* value. We additionally depict the *average* value in Table 2.

Overall, we observe that smaller values of k and larger values of ε tend to result in lower costs. However, the impact of ε on the noise is relatively minor compared to the effect of k. In fact, changing k has a noticeably stronger influence on the amount of noise introduced. That said, the noise remains small across all parameter settings.

While smaller k values help reduce costs, they also offer weaker privacy guarantees. Conversely, larger k improves privacy but increases the cost. Therefore, choosing an appropriate value for k requires balancing privacy and cost, depending on the specific requirements of the PPETC system.

Melbourne Case Study. The incurred costs for the Melbourne case study are depicted in Fig. 2 and Table 2, also using $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and $k \in \{3, 5, 7, 9, 11\}$. It is very notable that the mean of the cost is actually 0.00% for all ε and k. This is due to the distribution of possible wallets: While the Brisbane case study has 106 possible wallets, where each balance appears 1.14 times on average, the Melbourne case study has 285 possible wallets, where each balance appears ≈22 times on average (cp. Appendix B). Thus, it is highly likely that the k nearest neighbors of a wallet have the same wallet balance as the original wallet, resulting in no costs at all in most cases.

5.2 Evaluation of the Exponential Mechanism

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

5.2.1 ***Privacy Analysis.*** We evaluate the privacy level of an individual by calculating the success rate of the deobfuscation attack from Section 4.3, for different ε. To do so, for each obfuscated wallet from the set W_p, we compute the success rate of finding the associated original wallet from the set W_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 wallet, for different ε, α_{eucl}, and α_{sim}.

For ε, we will analyze $\varepsilon \in \{0.1, 0.5, 1, 2\}$. We will focus on $(\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25)$ in this section. In Appendix C, we will also evaluate for the parameters $(\alpha_{eucl} = 1, \alpha_{sim} = 0)$ and $(\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5)$. Since Appendix C shows that all three parameter configurations perform very similar, we chose to analyze $(\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25)$ in the main body as a balanced middle ground.

Brisbane Case Study. Figure 3 shows the success rate of the deobfuscation attack on the exponential mechanism for Brisbane. It can be noted that the “corner cases”, i.e., wallets with a balance close to \$0 or close to \$10, have a higher success chance than wallets with a balance “in the

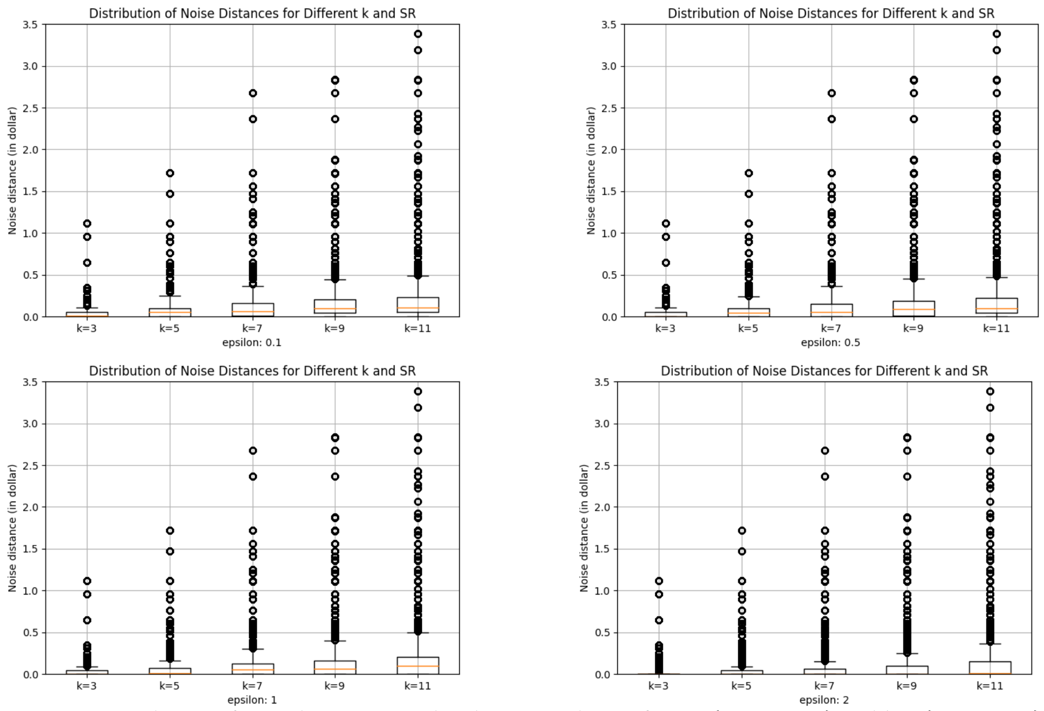


Fig. 1. KRR mechanism for Brisbane case study: The incurred costs for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and $k \in \{3, 5, 7, 9, 11\}$.

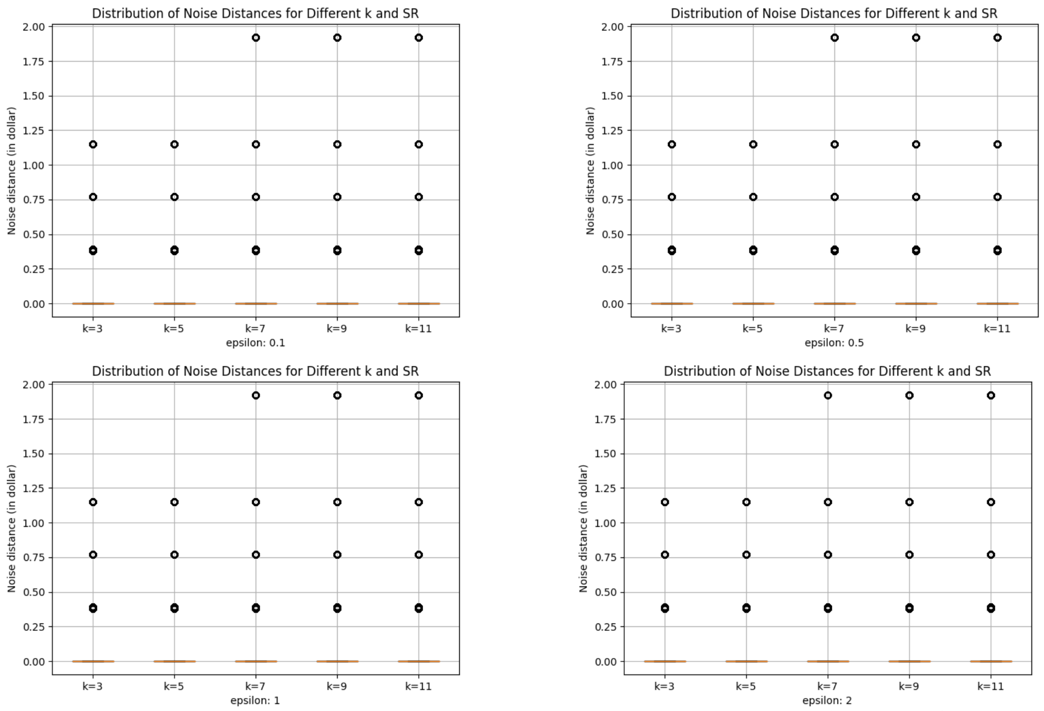


Fig. 2. KRR mechanism for Melbourne case study: The incurred costs for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and $k \in \{3, 5, 7, 9, 11\}$.

middle". But overall, for all εs the success rates are pretty low, i.e., between 0.9% − 1.5%. This is a stark contrast to the KRR mechanism, where the lowest success rate was 9.95%.

Melbourne Case Study. Figure 5 shows the success rate of the deobfuscation attack on the exponential mechanism for Melbourne. The same observations that were made for Brisbane (cp. Fig. 3) 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 the Melbourne case study, the success rated are between 0.3% − 0.7%, whereas they are between 0.9% − 1.5% for the Brisbane case study. This is probably due to the fact that the set of possible wallets W_p for Melbourne is more than two times as large as the one for Brisbane, thus the set of possible deobfuscated wallets is much larger.

5.2.2 ***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 wallet w_i, we obtain its associated obfuscated wallet w_o, using the exponential obfuscation mechanism (Algorithm 2).

Step 2: Having obtained the obfuscated wallet, we compute the cost as $|\textsc{get\_balance}(w_o) - \textsc{get\_balance}(w_i)|$.

We repeat this 1000 times to get a good estimate of the costs.

Brisbane Case Study. In Fig. 4, the incurred costs are shown for the Brisbane case study, for $\varepsilon \in \{0.1, 0.5, 1, 2\}$. The figure uses box plots to visualize the distribution of costs, where the yellow line inside each box indicates the *median* value. We additionally depict the *average* value in Table 2.

Overall, it can be seen that the costs decrease for larger ε. Also, the costs for the original wallets with a very small balance are significantly higher than the costs for all other wallets. This is due to the distribution of wallet balances (cp. Appendix B): For wallets with a large balance, there are many other wallets with a similar balance. But for wallets with a small balance, this is not the case. Thus, when obfuscating wallets with a very small balance, the chances to receive a more different obfuscated balance are higher.

Melbourne Case Study. The incurred costs for the Melbourne case study are depicted in Fig. 6 and Table 2, also using $\varepsilon \in \{0.1, 0.5, 1, 2\}$. The same observations that were made for the Brisbane case study can be made for the Melbourne case study as well, with the difference that all graphs have a bit lower costs here. While the Brisbane case study had costs averaging between \$1.89 − \$2.04, the Melbourne case study has costs averaging between \$1.71 − \$1.89.

6 Discussion

Discussion of Evaluation Results. We now compare the evaluation results of the KRR mechanism and the exponential mechanism across both case studies (Brisbane and Melbourne).

In terms of privacy, measured via the adversary's success rate, the exponential mechanism clearly outperforms KRR. The success rates for the exponential mechanism range between 0.35% and 1.5%, indicating strong privacy guarantees. In contrast, the KRR mechanism yields success rates between 10% and 78%, depending on the parameter configuration. This indicates that KRR generally provides less privacy than the exponential mechanism.

However, the situation is reversed when it comes to cost. The KRR mechanism incurs average costs ranging from \$0.02 to \$0.22, which is substantially lower than the exponential mechanism's costs, which range from \$1.71 to \$2.04. Therefore, the KRR mechanism is much more cost-efficient. Additionally, the exponential mechanism produces outliers with costs as high as \$8. Since the KRR mechanism only selects from the k nearest neighbors of w_i, rather than from the entire set of

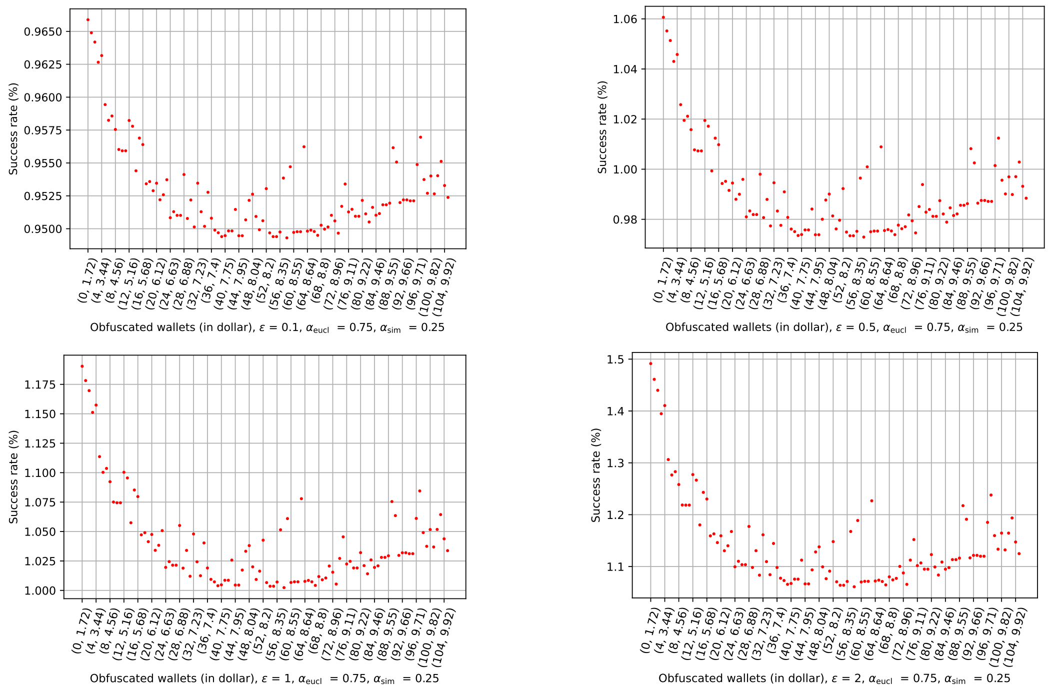


Fig. 3. Exponential mechanism for Brisbane Case Study: Each graph shows the success rate of deobfuscation for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and for $(\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25)$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

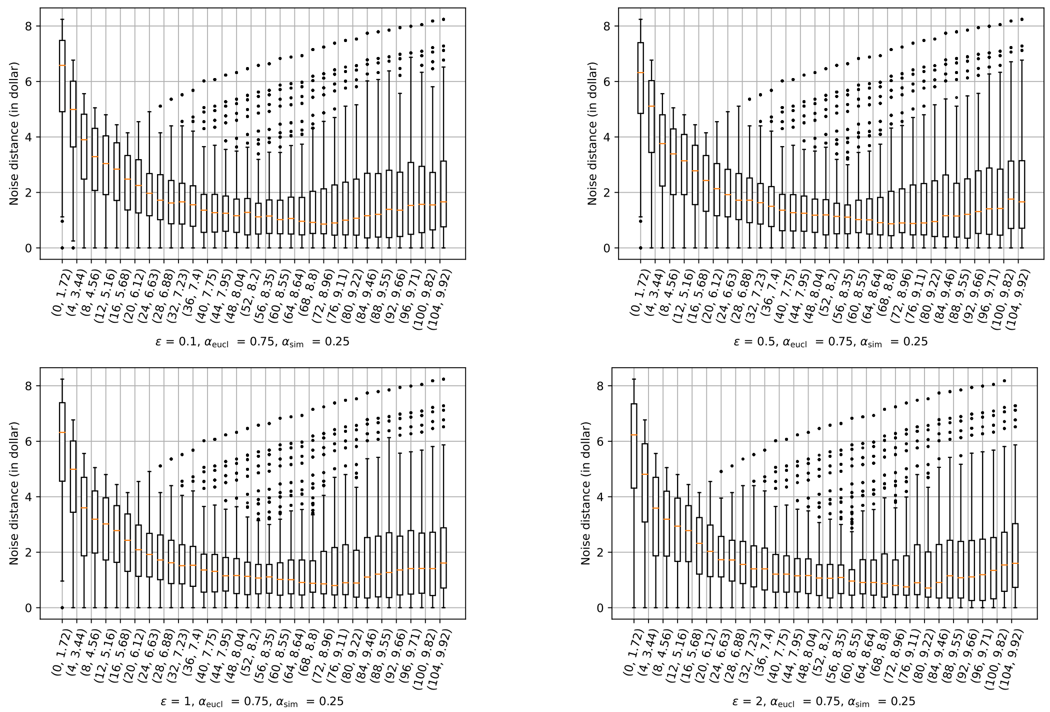


Fig. 4. Exponential mechanism for Brisbane case study: The incurred costs for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and $(\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25)$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

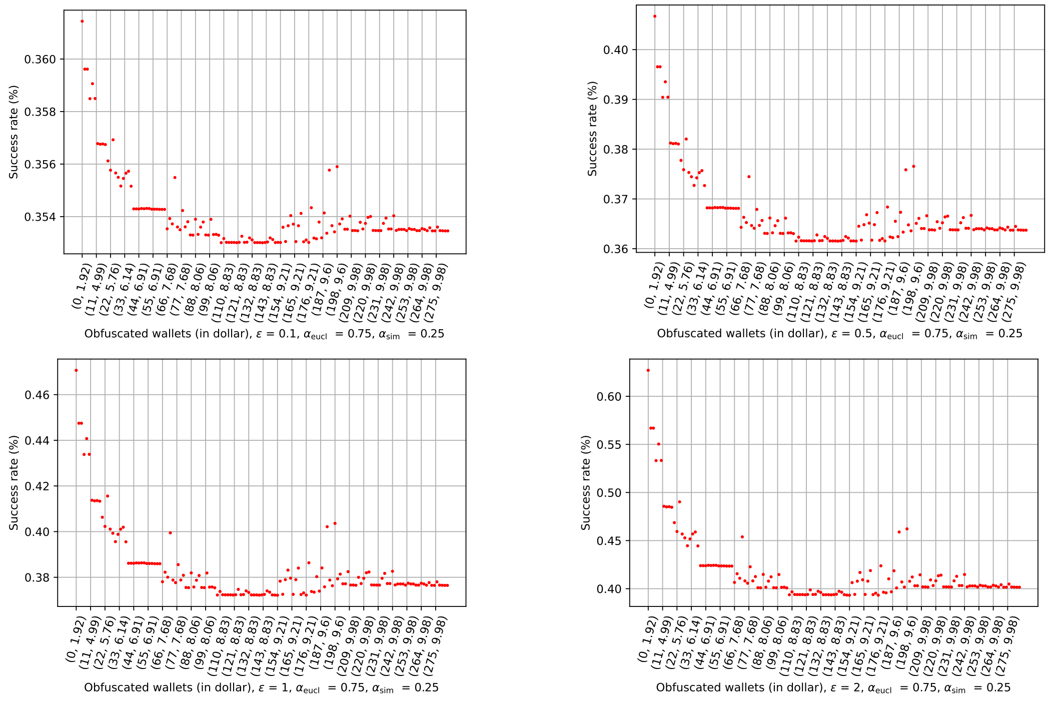


Fig. 5. Exponential mechanism for Melbourne Case Study: Each graph shows the success rate of deobfuscation for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and for $(\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25)$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

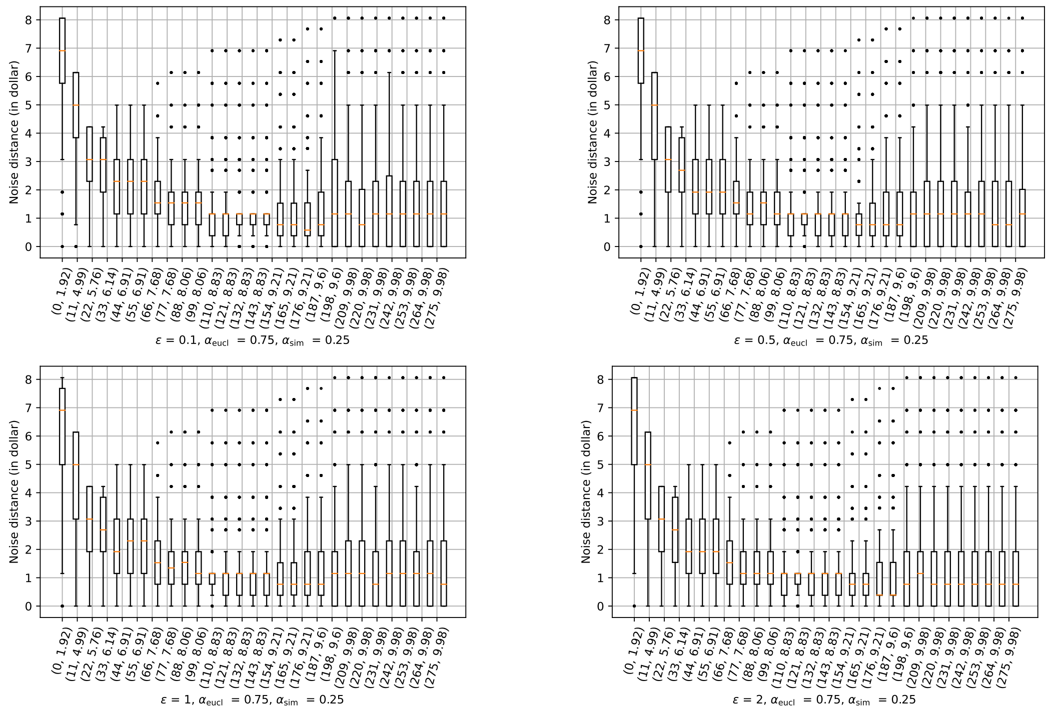


Fig. 6. Exponential mechanism for Melbourne case study: The incurred costs for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and $(\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25)$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

plausible wallets W_p, it does not produce these high outliers. KRR outliers are all below \$3, and the smaller k, the lower the highest outlier.

These results highlight a clear trade-off: the exponential mechanism provides better privacy, while KRR is more cost-efficient. Therefore, selecting an appropriate mechanism requires balancing these two aspects. A practical approach for a TSP is to first decide what level of privacy is acceptable, i.e., what maximum success rate is tolerable for an adversary. Once this threshold is defined, all mechanisms and parameter configurations that meet the privacy requirement can be identified. From this filtered set, the configuration with the lowest cost can then be selected.

Let us consider an example where a TSP decides that a maximum success rate of 15% is acceptable. Under this criteria, all configurations of the exponential mechanism are valid. For the KRR mechanism, the following configurations meet the privacy requirement: $(\varepsilon = 0.1, k = 9)$, $(\varepsilon = 0.1, k = 11)$, and $(\varepsilon = 0.5, k = 11)$. Among these, the configuration $(\varepsilon = 0.1, k = 9)$ yields the lowest costs. It has an average cost of \$0.18 (with a mean of \$0.10 for the Brisbane case study and a mean of \$0.00 for the Melbourne case study) and a success rate of 12.14%. This would make it a strong candidate for deployment.

Discussion of the Adversarial Model. It should be noted that while our evaluation focuses on a relatively weak adversarial model, more sophisticated adversaries may employ more advanced strategies, which potentially achieve higher success rates. Defending against such stronger attacks would require mechanisms that induce greater obfuscation, which in turn leads to increased costs for the user. Consequently, the use of LDP mechanisms is not optional for PPETC systems but a necessary safeguard to ensure protection against at least the baseline adversary, whose attack strategies are both plausible and likely to be executed in practice. Even these foundational mechanisms impose non-negligible costs, underscoring that privacy in this context is not free.

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 larger additional cost. Thus, the TSP could offer some kind of *reimbursement mechanism*, like the one in [11], 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 wallets with a higher probability than if only one month was considered. For example, 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 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 [9]. 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 we use in Section 5 is vulnerable to attacks: For wallet balances ≤10 AUD, the adversary could identify the visited toll stations with a 94% success rate by solving the SSP problem. [9] also shows that solving the SSP helps to effectively recover user traces. In contrast to [2], the adversary in [9] uses every information the TSP gets.

Protection Mechanisms. Some PPETC schemes [4, 16, 27] 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 [11] 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. [11] additionally proposes a cryptographic protocol that helps customers reclaim the additional expenditure incurred for the sake of privacy. A limitation of [11] 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 [17] 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

Previous work has shown that a PPETC version of the Brisbane ETC system is vulnerable to trace recovery attacks, with success rates reaching up to 94%. In this work, we investigated whether common ε-LDP mechanisms, specifically k-ary randomized response (KRR) and the exponential mechanism, can help mitigate these attacks.

Our evaluation for two real-world case studies shows that both mechanisms significantly reduce the adversary's success rate. The exponential mechanism achieves very strong privacy, with success rates between 0.35% and 1.5% across both case studies. However, this comes at a cost: users may pay up to \$2.04 extra (on average) in a single billing period, which could discourage adoption. In contrast, the KRR mechanism is much more cost-efficient, with average costs between \$0.02 and \$0.22, but its privacy protection is weaker, with success rates ranging from 10% to 78%.

This highlights a clear trade-off between privacy and cost. It is up to the TSP (probably with the involvement of data protection lawyers) to decide what level of privacy is acceptable, i.e., what maximum success rate is tolerable. Once this threshold is defined, suitable mechanisms and parameter configurations can be selected accordingly. For example, if a 15% success rate is acceptable, the KRR mechanism with $(\varepsilon = 0.1, k = 9)$ offers a good balance, with a 12.14% success rate and \$0.28 average cost.

In summary, both mechanisms are viable options for improving privacy in PPETC systems, but the final choice depends on the privacy-cost balance that the TSP is willing to accept.

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] Manuel Blum. 1983. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News* 15, 1 (1983), 23–27. doi:10.1145/1008908.1008911

[7] Linkt Brisbane. 2022. *Queensland toll calculator*. Retrieved 2022 from https://www.linkt.com.au/using-toll-roads/toll-calculator/brisbane

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Cynthia Dwork. 2006. Differential privacy. In *International colloquium on automata, languages, and programming*. Springer, 1–12.

[13] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*. Springer, 1–19.

[14] Cynthia Dwork, Nitin Kohli, and Deirdre Mulligan. 2019. Differential privacy in practice: Expose your epsilons! *Journal of Privacy and Confidentiality* 9, 2 (2019).

[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] 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).

[17] 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.

[18] 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.

[19] 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).

[20] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. 2016. Discrete Distribution Estimation under Local Privacy. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 2436–2444. https://proceedings.mlr.press/v48/kairouz16.html

[21] 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.

[22] Florian Kerschbaum and Hoon Wei Lim. 2015. Privacy-preserving observation in public spaces. In *European Symposium on Research in Computer Security*. Springer, 81–100.

[23] Jeffrey C Lagarias and Andrew M Odlyzko. 1985. Solving low-density subset sum problems. *Journal of the ACM (JACM)* 32, 1 (1985), 229–246.

[24] Linkt. 2024. *Toll pricing*. Retrieved 2024 from https://www.linkt.com.au/using-toll-roads/about-toll-roads/citylink/toll-pricing/melbourne

[25] 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.

[26] KW Ogden. 2001. Privacy issues in electronic toll collection. *Transportation Research Part C: Emerging Technologies* 9, 2 (2001), 123–134.

[27] Raluca Ada Popa, Hari Balakrishnan, and Andrew J Blumberg. 2009. VPriv: Protecting privacy in location-based vehicular services. (2009).

[28] Likun Qin, Nan Wang, and Tianshuo Qiu. 2023. A Survey of Local Differential Privacy and Its Variants. arXiv:2309.00861 [cs.CR] https://arxiv.org/abs/2309.00861

[29] Reza Shokri. 2014. Privacy games: Optimal user-centric data obfuscation. *arXiv preprint arXiv:1402.3426* (2014).

[30] Reza Shokri, George Theodorakopoulos, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. 2012. Protecting location privacy: optimal strategy against localization attacks. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 617–627.

[31] 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

[32] Transurban. 2022. *Travel on our roads.* Retrieved 2022 from https://insights.transurban.com/travel/travel-on-our-roads/#toll-spend-data

[33] 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.

[34] Stanley L. Warner. 1965. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *J. Amer. Statist. Assoc.* 60, 309 (1965), 63–69. doi:10.1080/01621459.1965.10480775 PMID: 12261830.

[35] 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.

A Details of the Exponential Obfuscation Mechanism

In this appendix, we explain the Exponential obfuscation mechanism (Algorithm 2) in more detail, as well as Algorithms 5, 6, 7 and 8, which are sub-algorithms of Algorithm 2.

A.1 Algorithm EXPONENTIAL_OBFUSCATION

Algorithm 2 takes as input $(w_i \in W_p, W_p, \varepsilon, \alpha_{eucl} \in [0,1], \alpha_{sim} \in [0,1])$ and outputs the obfuscated wallet w_o. 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 5) in line 4. Note that the precomputation step needs only to be executed once, as long as W_p stays the same.

Step 2: Score Calculation. Given an input wallet w_i, we now compute the score of the input wallet's trace $trace$ and every possible trace in lines 5 to 13, 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 6). We also compute the similarity distance d_{sim} between the traces with COMPUTE_SIMILARITY (Algorithm 7). Afterwards we scale d_{eucl} to the range $[0,1]$ by dividing it by the maximum possible Euclidean distance (calculated in line 4). 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 14 and 15. According to the Exponential mechanism, we calculate the the probability as specified in Eq. (6). We calculate the numerator of Eq. (6) in line 14 and then apply normalization in line 15.

Algorithm 5 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 6 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
```

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 16 an output trace $trace_k$ using the previously calculated probability distribution. Finally, we output the wallet corresponding to $trace_k$ as obfuscated wallet w_o.

A.2 Algorithm COMPUTE_MAX_DIST

In Algorithm 5 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.

A.3 Algorithm COMPUTE_EUCLIDEAN

Algorithm 6 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 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 $euclidean\_dist$.

Algorithm 7 Compute Similarity Distance

```
Input: $trace_1 \in T_p, trace_2 \in T_p$
Output: $sim\_dist$
1: function COMPUTE_SIMILARITY($trace_1, trace_2$)
2:     $sim\_dist \leftarrow 0$
3:     for all $(s_j, f_j) \in trace_1$ and $(s_j, f'_j) \in trace_2$ do
4:         if $f_j \neq 0$ and $f'_j \neq 0$ then
5:             $sim\_dist \leftarrow sim\_dist + (f_j - f'_j)^2$
6:         else if $f_j \neq 0$ and $f'_j == 0$ then
7:             $sim\_dist \leftarrow sim\_dist + (f_j)^2 + penalty^2$
8:         else if $f_j == 0$ and $f'_j \neq 0$ then
9:             $sim\_dist \leftarrow sim\_dist + (f'_j)^2 + penalty^2$
10:        end if
11:    end for
12:    $sim\_dist \leftarrow \sqrt{sim\_dist}$
13:    return $sim\_dist$
14: end function
```

Algorithm 8 Compute Probabilities

```
Input: $arr\_score, \varepsilon, \Delta$
Output: $arr\_prob$
1: function COMPUTE_PROB($arr\_score, \varepsilon, \Delta$)
2:     Declare $arr\_prob[|arr\_score|]$
3:     for all $score_i \in arr\_score$ do
4:         $arr\_prob[i] \leftarrow e^{(\varepsilon \cdot score_i)/(2 \cdot \Delta)}$
5:     end for
6:     return $arr\_prob$
7: end function
```

A.4 Algorithm COMPUTE_SIMILARITY

Algorithm 7 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 1 (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.

A.5 Algorithm compute_prob

Algorithm 8 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. (6)) and stores it in the list *arr_prob*. Finally, the algorithm returns *arr_prob* as output.

B Details of the Case Studies

Brisbane. Brisbane has the following list of possible wallets below \$10:

$W_p :=$ {1.72, 2.68, 2.84, 3.19, 3.44, 4.09, 4.4, 4.55, 4.56, 4.91, 5.11, 5.11, 5.16, 5.36, 5.46, 5.52, 5.68, 5.81, 5.87, 6.03, 6.12, 6.27, 6.28, 6.38, 6.63, 6.77, 6.83, 6.83, 6.88, 6.93, 7.08, 7.18, 7.23, 7.24, 7.28, 7.39, 7.4, 7.53, 7.59, 7.74, 7.75, 7.79, 7.79, 7.84, 7.95, 7.95, 7.99, 8.0, 8.04, 8.1, 8.14, 8.18, 8.2, 8.3, 8.3, 8.3, 8.35, 8.36, 8.49, 8.52, 8.55, 8.55, 8.55, 8.6, 8.64, 8.65, 8.65, 8.71, 8.8, 8.87, 8.9, 8.95, 8.96, 9.0, 9.06, 9.1, 9.11, 9.12, 9.2, 9.2, 9.22, 9.25, 9.31, 9.45, 9.46, 9.47, 9.51, 9.51, 9.55, 9.56, 9.57, 9.61, 9.66, 9.66, 9.67, 9.67, 9.71, 9.72, 9.76, 9.77, 9.82, 9.86, 9.9, 9.91, 9.92, 9.96}

This list has length of 106 entries. It can be noted that some wallet balances appear multiple times in the list. On average, each wallet balance appears 1.14 times.

Melbourne. Melbourne has the following list of possible wallets below \$10:

$W_p :=$ {1.92, 1.92, 3.07, 3.07, 3.07, 3.07, 3.84, 3.84, 3.84, 3.84, 3.84, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 5.76, 5.76, 5.76, 5.76, 5.76, 5.76, 5.76, 5.76, 6.14, 6.14, 6.14, 6.14, 6.14, 6.14, 6.14, 6.14, 6.14, 6.14, 6.14, 6.14, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 6.91, 7.68, 7.68, 7.68, 7.68, 7.68, 7.68, 7.68, 7.68, 7.68, 7.68, 7.68, 7.68, 7.68, 7.68, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.06, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 8.83, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.21, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.6, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98, 9.98}

This list has length of 285 entries. It can be noted that *every* wallet balance appears multiple times in the list. On average, each wallet balance appears ≈ 22 times.

C 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 consider $(\alpha_{eucl} = 1, \alpha_{sim} = 0)$ and $(\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5)$.

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

The privacy analysis for Brisbane and Melbourne can be found in Fig. 7 and Fig. 9, respectively. The cost analysis for Brisbane can be found in Fig. 8 and the one for Melbourne in Fig. 10. Also, the average costs across all wallets are included in Table 3.

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

The privacy analysis for Brisbane and Melbourne can be found in Fig. 11 and Fig. 13, respectively. The cost analysis for Brisbane can be found in Fig. 12 and the one for Melbourne in Fig. 14. Also, the average costs across all wallets are included in Table 3.

C.3 Privacy and Cost Analysis

When considering *privacy*, the success rates for $(\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5)$ in Figs. 11 and 13 and $(\alpha_{eucl} = 1, \alpha_{sim} = 0)$ in Figs. 7 and 9 do not differ much from the success rates of $(\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25)$ in Figs. 3 and 5. While $(\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5)$ yields slightly lower success rates than $(\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25)$, and those are slightly lower than $(\alpha_{eucl} = 1, \alpha_{sim} = 0)$, the differences are pretty much negligible. This applies to both case studies.

For *cost*, a similar pattern emerges. The cost for $(\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5)$ in Figs. 12 and 14 and $(\alpha_{eucl} = 1, \alpha_{sim} = 0)$ in Figs. 8 and 10 do not differ much from the success rates of $(\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25)$ in Figs. 4 and 6. Here, $(\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5)$ results in slightly higher costs than $(\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25)$, which in turn are slightly higher than those for $(\alpha_{eucl} = 1, \alpha_{sim} = 0)$. Again, the differences are very small and apply to both case studies.

In summary, both the success rates and the associated costs are very similar across all three parameter configurations. Since none of them clearly outperforms the others, we chose to analyze $(\alpha_{eucl} = 0.75, \alpha_{sim} = 0.25)$ in the main body as a balanced middle ground.

D Evaluation of the Laplace Mechanism

In this appendix, we evaluate the classical Laplace mechanism using our two case studies.

For obfuscation, we apply the Laplace mechanism as defined in Definition 2.2. The sensitivity in this setting corresponds to the difference between the highest and lowest possible wallet balances within the considered range [\$0, \$10]. For the Brisbane case study, the sensitivity is 8.24, and for the Melbourne case study it is 8.06 (cp. Appendix B).

To evaluate deobfuscation, we again assume an adversary following a maximum likelihood strategy. Given an obfuscated wallet, the adversary identifies the wallet balance with the highest likelihood of being the original (based on the Laplace distribution's probability density function). Among all wallets with that balance, one is selected at random and returned as the adversary's guess.

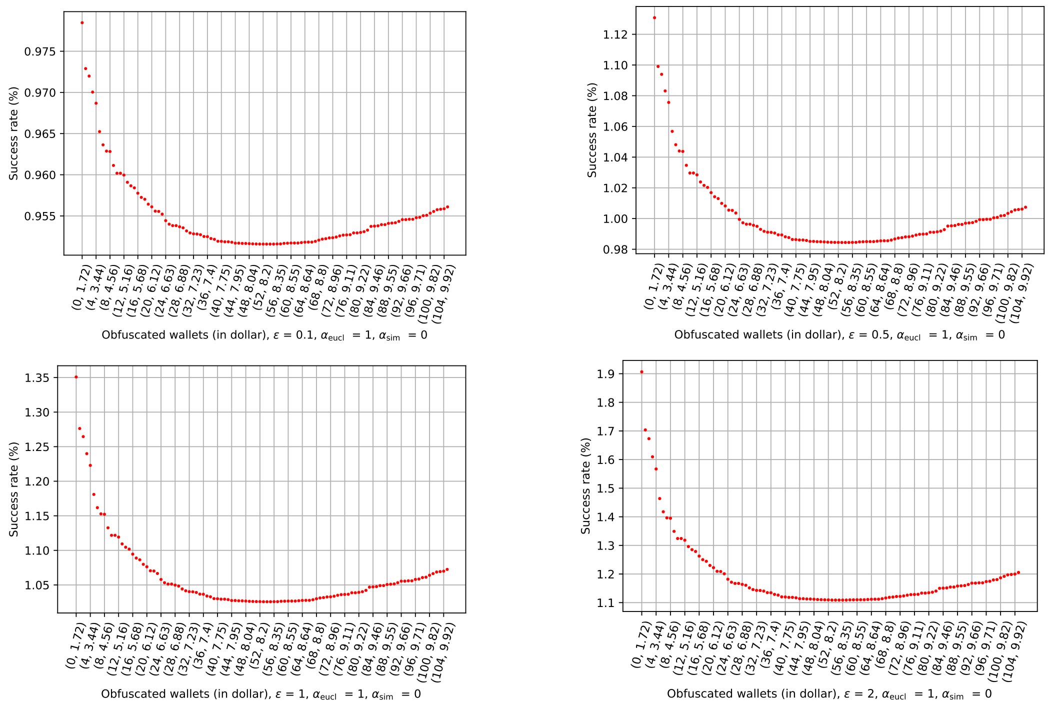


Fig. 7. Exponential mechanism for Brisbane Case Study: Each graph shows the success rate of deobfuscation for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and for $(\alpha_{eucl} = 1, \alpha_{sim} = 0)$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

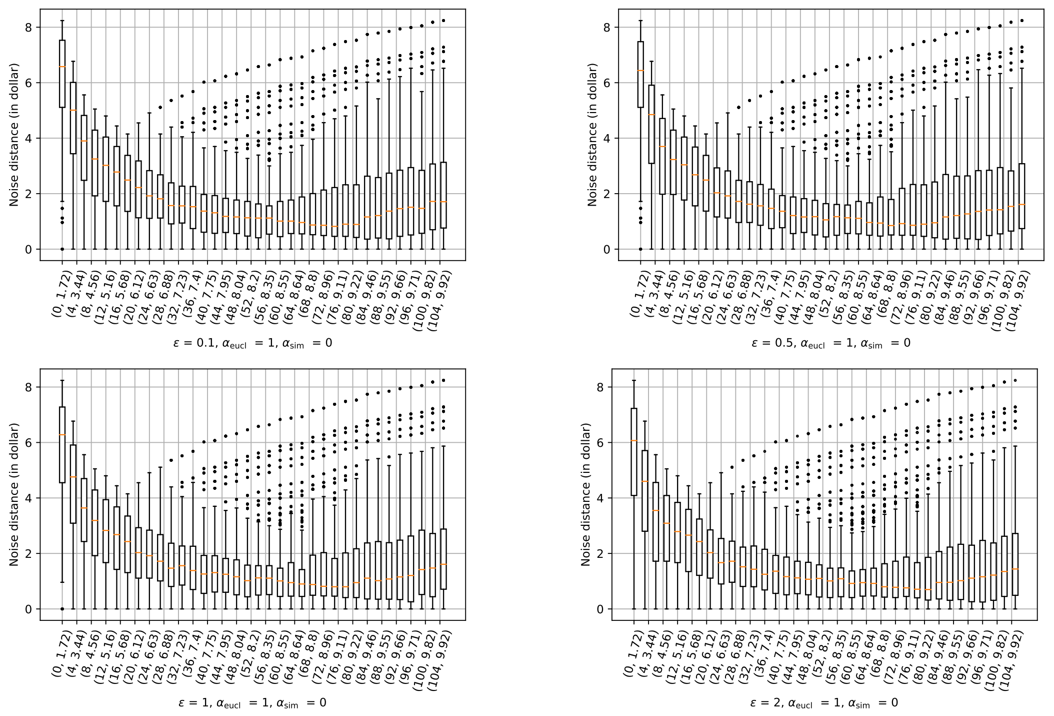


Fig. 8. Exponential mechanism for Brisbane case study: The incurred costs for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and $(\alpha_{eucl} = 1, \alpha_{sim} = 0)$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

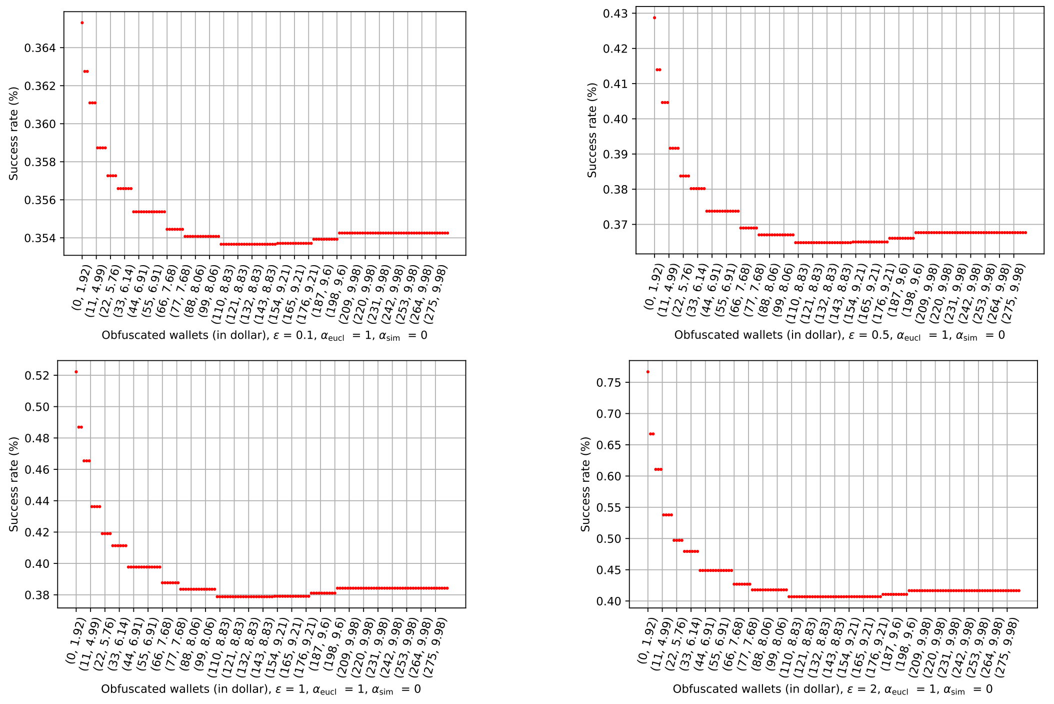


Fig. 9. Exponential mechanism for Melbourne Case Study: Each graph shows the success rate of deobfuscation for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and for $(\alpha_{eucl} = 1, \alpha_{sim} = 0)$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

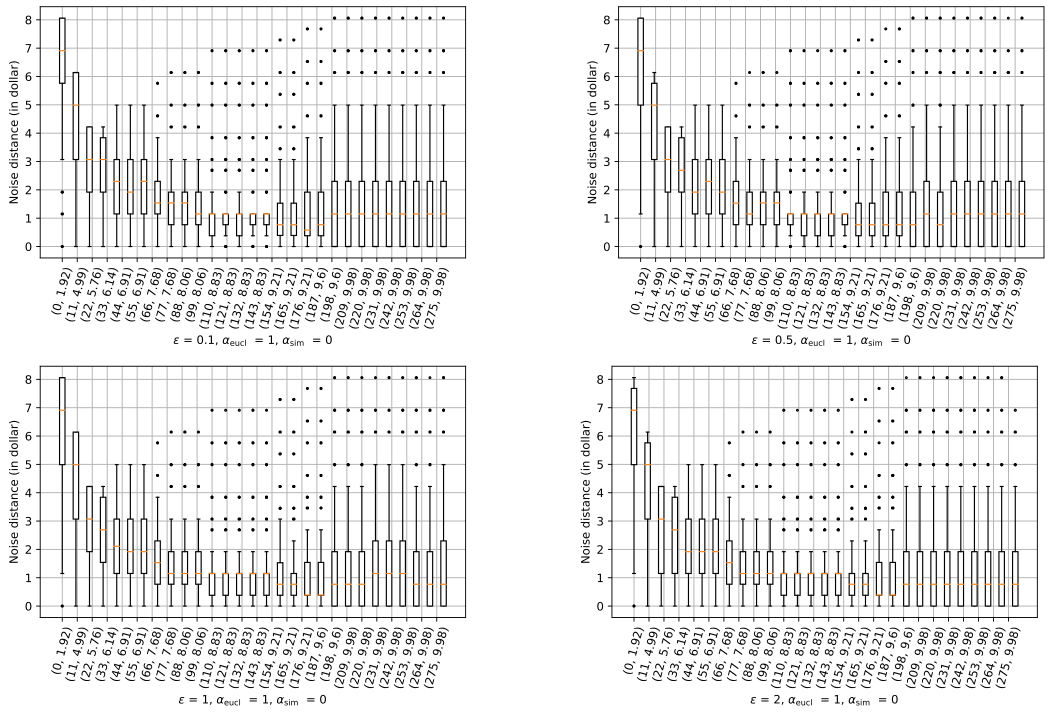


Fig. 10. Exponential mechanism for Melbourne case study: The incurred costs for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and $(\alpha_{eucl} = 1, \alpha_{sim} = 0)$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

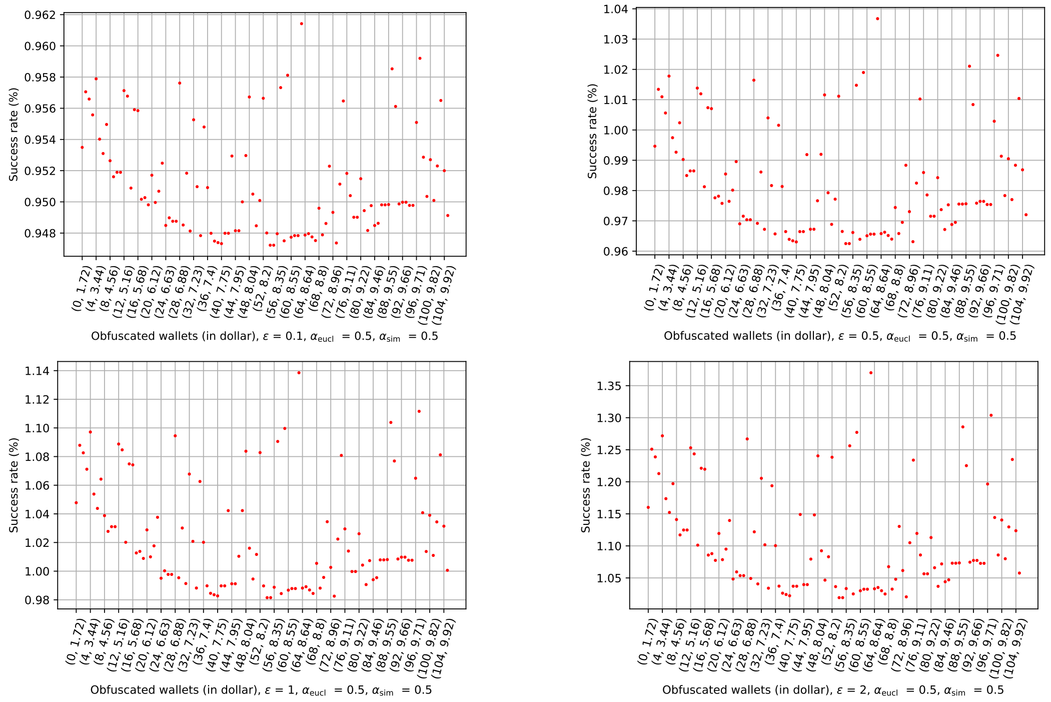


Fig. 11. Exponential mechanism for Brisbane Case Study: Each graph shows the success rate of deobfuscation for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and for $(\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5)$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

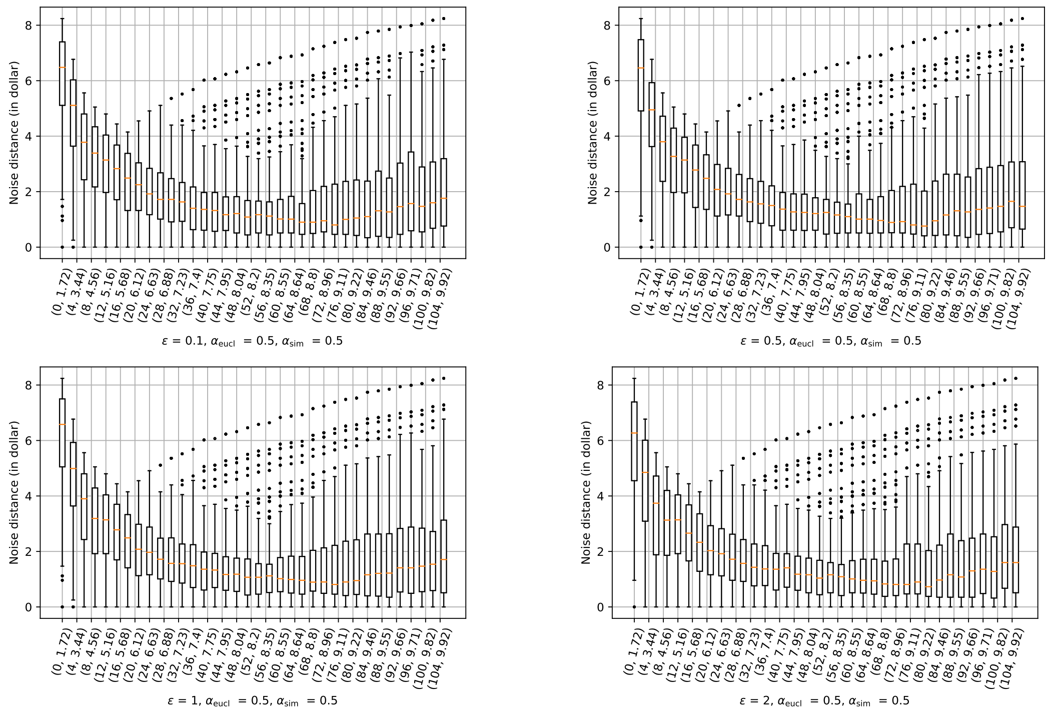


Fig. 12. Exponential mechanism for Brisbane case study: The incurred costs for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and $(\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5)$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

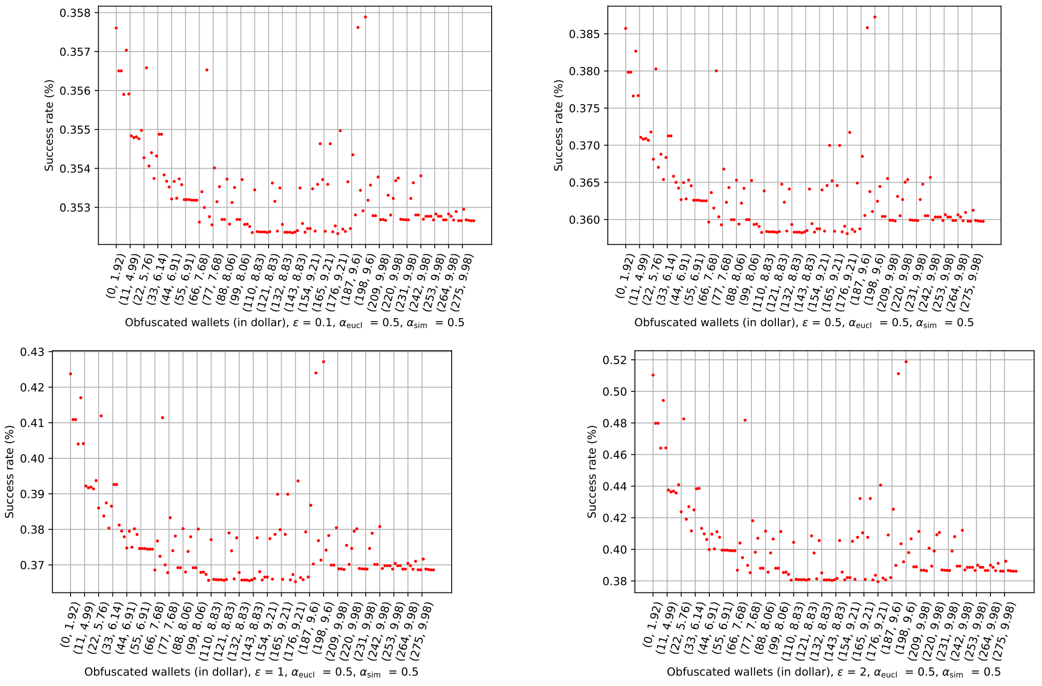


Fig. 13. Exponential mechanism for Melbourne Case Study: Each graph shows the success rate of deobfuscation for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and for $(\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5)$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

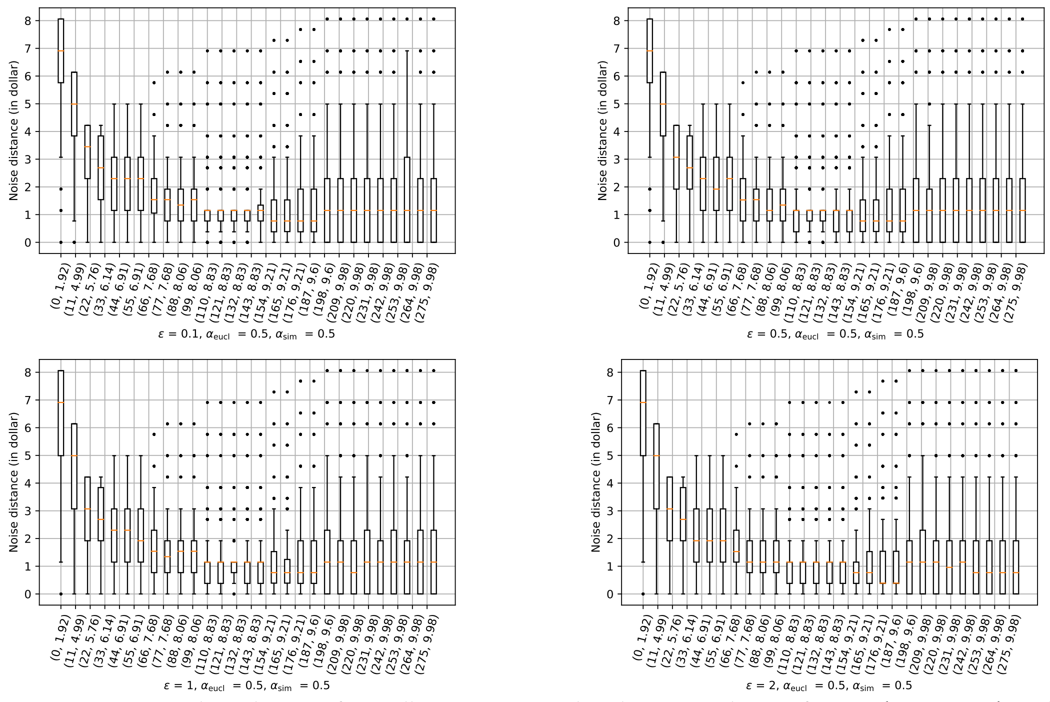


Fig. 14. Exponential mechanism for Melbourne case study: The incurred costs for $\varepsilon \in \{0.1, 0.5, 1, 2\}$ and $(\alpha_{eucl} = 0.5, \alpha_{sim} = 0.5)$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

	Brisbane									Melbourne								
	KRR with k =					EXP with α_{eucl} =			LAP	KRR with k =					EXP with α_{eucl} =			LAP
ε	3	5	7	9	11	1	0.75	0.5		3	5	7	9	11	1	0.75	0.5	
0.1	0.05	0.09	0.14	0.18	0.22	2.05	2.04	2.05	4.78	0.02	0.03	0.05	0.06	0.08	1.88	1.89	1.86	4.76
0.5	0.04	0.09	0.13	0.17	0.21	2.00	2.01	2.04	4.06	0.01	0.03	0.04	0.05	0.06	1.82	1.83	1.84	4.07
1	0.03	0.07	0.11	0.15	0.19	1.94	1.96	1.99	3.40	0.01	0.02	0.03	0.03	0.04	1.75	1.80	1.80	3.44
2	0.02	0.04	0.07	0.10	0.14	1.82	1.89	1.95	2.47	0.00	0.01	0.01	0.02	0.02	1.65	1.71	1.77	2.51

Table 3. Average cost of obfuscation in AUD across *all* mechanisms examined in this paper, for different ε.

Evaluation Results for the Laplace Mechanism. The success rates of the deobfuscation attack are shown in Fig. 15 for the Brisbane case study and in Fig. 17 for the Melbourne case study. The corresponding incurred costs are depicted in Fig. 16 and Fig. 18, respectively. Table 3 summarizes the average costs across all wallet balances for all evaluated mechanisms.

In terms of privacy, the results show generally low success rates for the adversary across all considered values of ε. However, each scenario includes one outlier wallet with a significantly higher chance of being correctly deobfuscated, reaching up to 55% success probability.

Regarding cost, the average incurred costs range from \$2.47 to \$4.78, depending on the privacy parameter.

Comparison to KRR and the Exponential Mechanism. Compared to the mechanisms discussed in the main body of the paper, namely KRR and the exponential mechanism, the Laplace mechanism does not outperform either in any category. While both the Laplace and exponential mechanisms offer strong privacy protection, the Laplace mechanism suffers from outlier wallets with significantly higher success rates. In contrast, the exponential mechanism maintains consistently low success rates across all wallets. In terms of cost, KRR is the most efficient and also exhibits no outliers. It consistently achieves low costs while still providing a reasonable level of privacy. The Laplace mechanism incurs lower costs than the exponential mechanism but remains less efficient than KRR.

In summary, the Laplace mechanism offers neither the strongest privacy guarantees nor the lowest cost. For this reason, we chose to present its evaluation in the appendix rather than in the main body of the paper.

E 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 E.1 how to obtain the *average similarity of traces*. In Appendix E.2 we then provide the algorithms for the similar trace attack. Finally, in Appendix E.3 we evaluate the similar trace attack.

E.1 Average Similarity Between Traces

In Algorithm 9, 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 7. Afterwards we calculate the average *avg* over all the differences.

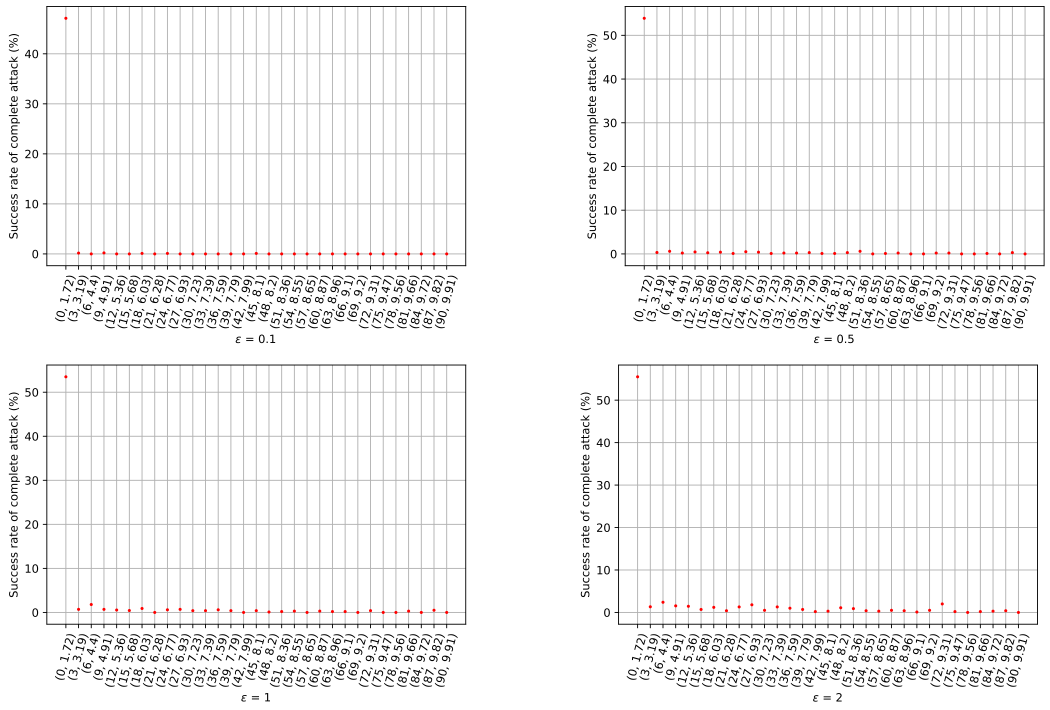


Fig. 15. Laplace mechanism for Brisbane Case Study: Each graph shows the success rate of deobfuscation for $\varepsilon \in \{0.1, 0.5, 1, 2\}$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

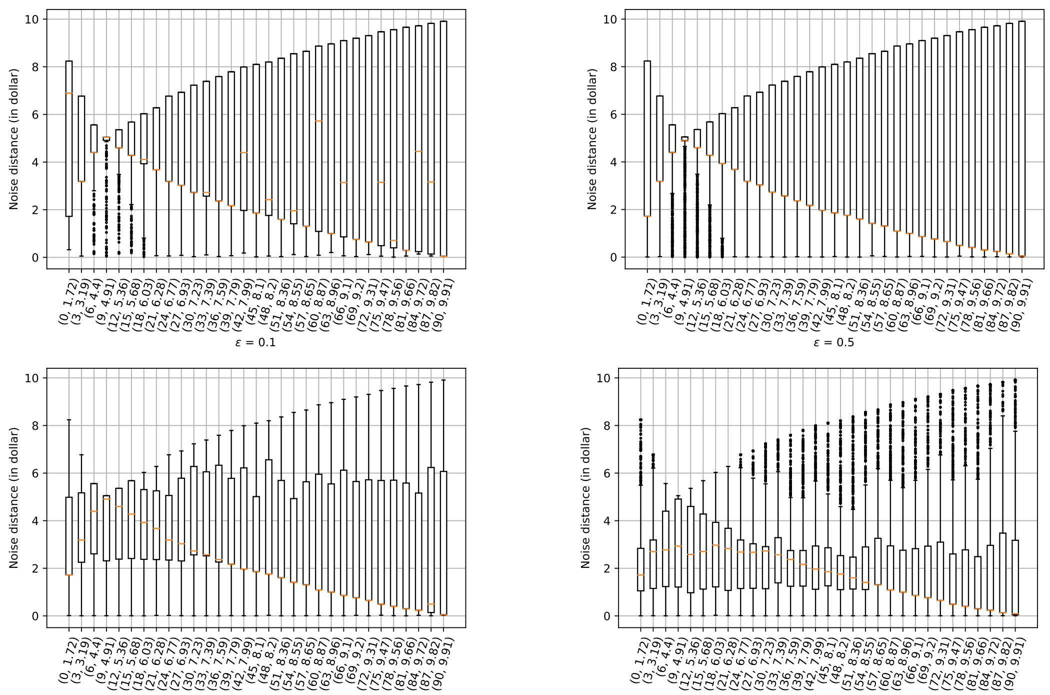


Fig. 16. Laplace mechanism for Brisbane case study: The incurred costs for $\varepsilon \in \{0.1, 0.5, 1, 2\}$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

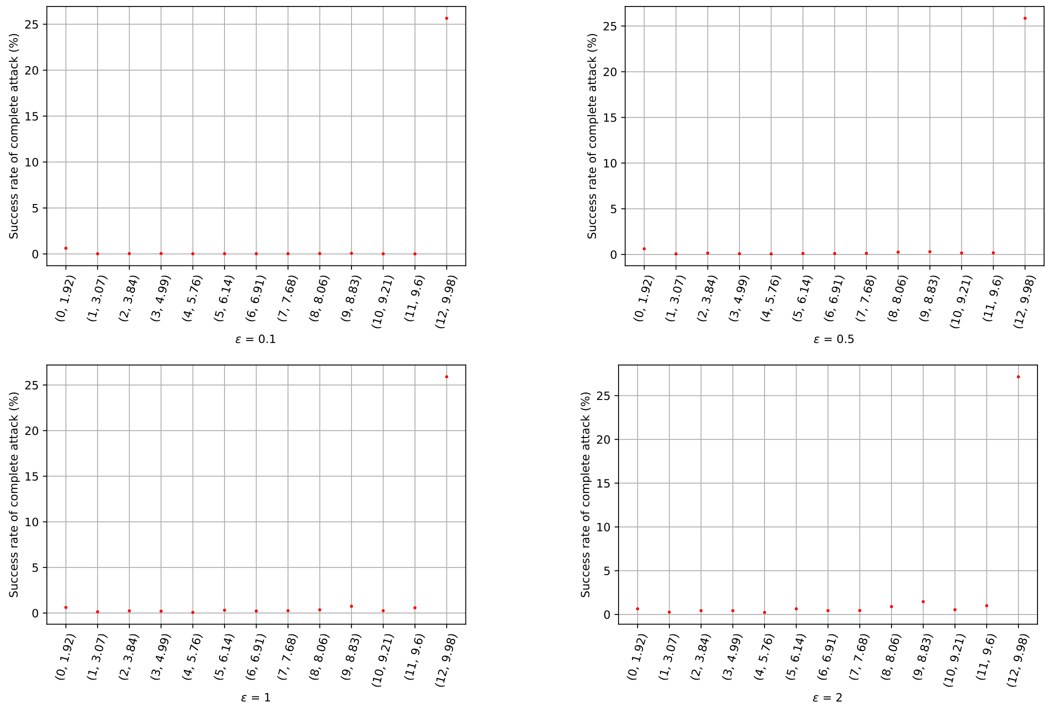


Fig. 17. Laplace mechanism for Melbourne Case Study: Each graph shows the success rate of deobfuscation for $\varepsilon \in \{0.1, 0.5, 1, 2\}$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

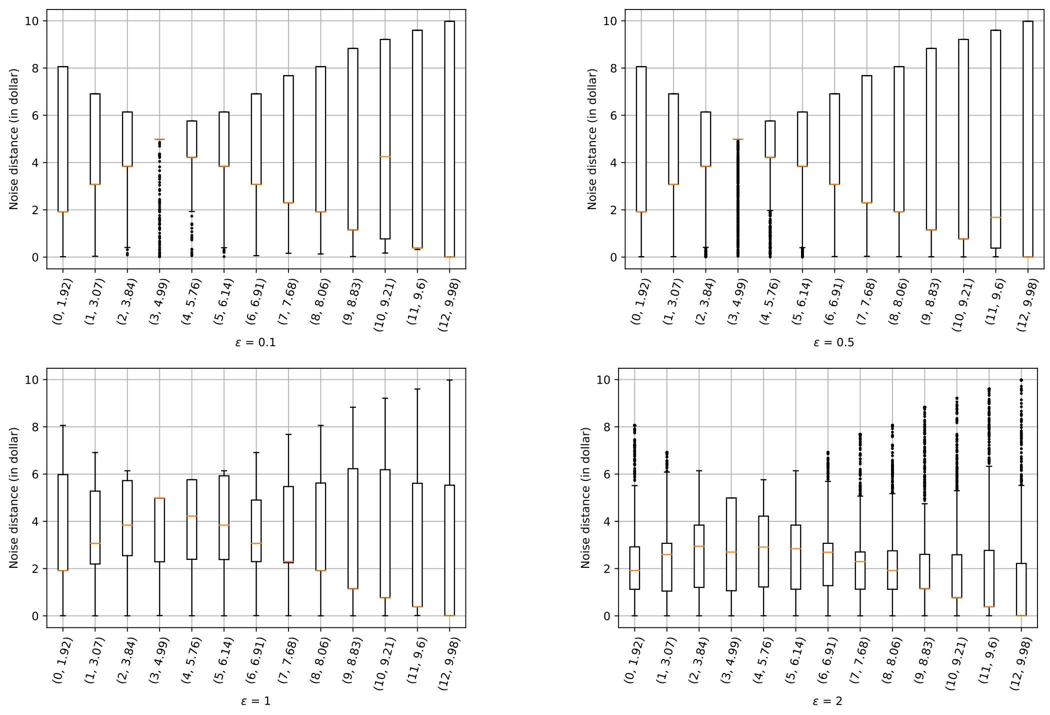


Fig. 18. Laplace mechanism for Melbourne case study: The incurred costs for $\varepsilon \in \{0.1, 0.5, 1, 2\}$. The tuples on the x-axis indicate a wallet $(id, bal, \cdot)$.

E.2 Executing the Similar Trace Attack

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

The similar trace attack for KRR is depicted in Algorithm 10. The similar trace attack for the exponential mechanism is depicted in Algorithm 11. Note that we here assume that the precomputation algorithm of the Exponential deobfuscation algorithm (cp. Algorithm 3) has been executed beforehand.

E.3 Evaluating the Similar Trace Attack

We describe the results of the similar trace attack for both KRR and the exponential mechanism.

E.3.1 ***Evaluation Approach.***

KRR. Our evaluation of the similar trace attack for KRR proceeds as follows:

(1) Fix parameters W_p, k, ε, num.
(2) For each $w_i \in W_p$, we execute KRR_RELAXED_ATK$(w_i, W_p, k, \varepsilon, num)$ to get a list arr_d_{sim}. KRR_RELAXED_ATK basically obfuscates the original wallet w_i, then deobfuscates it again and calculates the similarity between the deobfuscated wallet's trace and the original wallet's trace. This is repeated *num* times and the similarity after each deobfuscation is stored in an element of arr_d_{sim}.
(3) The results are plotted in a graph, where each element on the x-axis equals an original wallet 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 wallet.

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

(1) Fix parameters W_p, ε, α_{eucl}, α_{sim}, num.
(2) Precomputation: Execute PRECOMPUTATION_EXP_ATTACK$(W_p, \varepsilon, \alpha_{eucl}, \alpha_{sim})$ to get the table $table$ with all probabilities.
(3) For each $w_i \in W_p$, execute EXP_RELAXED_ATK$(w_i, W_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}, num, table)$ to get a list arr_d_{sim}. EXP_RELAXED_ATK basically obfuscates the original wallet w_i, then deobfuscates it again and calculates the similarity between the deobfuscated wallet's trace and the original wallet's trace. 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 wallet 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 wallet.

E.3.2 ***Results.*** In Fig. 19 the results of $num = 1000$ iterations of the similar trace attack are depicted, in Fig. 19a for KRR and in Fig. 19b for the exponential mechanism. For both mechanism it can be seen that this attack does not fare much better than just guessing a random wallet. Since the yellow/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 wallet

Algorithm 9 Compute Average Similarity Between Traces

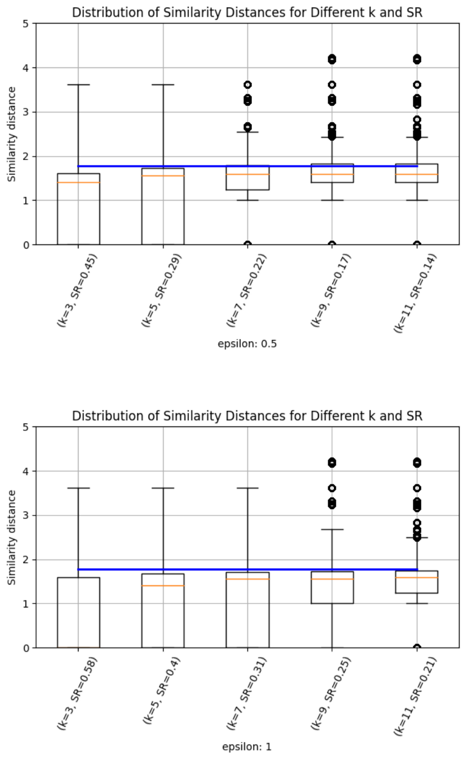
```
Input: $T_p$
Output: $avg$
 1: function COMPUTE_AVERAGE_SIMILARITY($T_p$)
 2:     Initialize $arr\_d_{sim}$ as empty list
 3:     for all $trace_i \in T_p$ do
 4:         for all $trace_j \in T_p$ do
 5:             $d_{sim} \leftarrow$ COMPUTE_SIMILARITY($trace_i, trace_j$)
 6:             Append $d_{sim}$ to $arr\_d_{sim}$
 7:         end for
 8:     end for
 9:     $avg \leftarrow$ AVERAGE($arr\_d_{sim}$)          \\Compute average of list
10:     return $avg$
11: end function
```

Algorithm 10 Similar Trace Attack for KRR

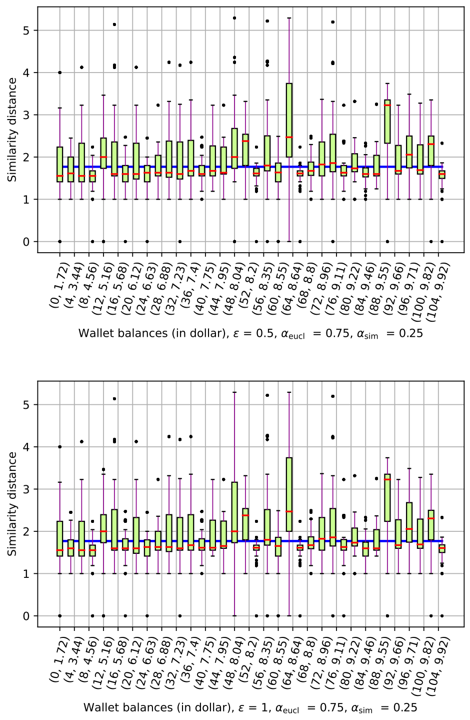
```
Input: $w_i \in W_p, W_p, k \in \{1, \ldots, |W_p|\}, \varepsilon > 0, num \in \mathbb{N}$
Output: $arr\_d_{sim}$
 1: function KRR_RELAXED_ATK($w_i, W_p, k, \varepsilon, num$)
 2:     Initialize $arr\_d_{sim}$ as empty list
 3:     for $i$ from 1 to $num$ do
 4:         $w_o \leftarrow$ KRR_OBFUSCATION($w_i, W_p, k, \varepsilon$)
 5:         $w_d \leftarrow w_o$          \\Deobfuscation just selects obfuscated wallet
 6:         Parse $(\cdot, \cdot, trace_i) \leftarrow w_i$
 7:         Parse $(\cdot, \cdot, trace_d) \leftarrow w_d$
 8:         $d_{sim} \leftarrow$ COMPUTE_SIMILARITY($trace_i, trace_d$)
 9:         Append $d_{sim}$ to $arr\_d_{sim}$
10:     end for
11:     return $arr\_d_{sim}$
12: end function
```

Algorithm 11 Similar Trace Attack for Exponential

```
Input: $w_i \in W_p, W_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}, num \in \mathbb{N}, table$
Output: $arr\_d_{sim}$
 1: function EXP_RELAXED_ATK($w_i, W_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}, num, table$)
 2:     Initialize $arr\_d_{sim}$ as empty list
 3:     for $i$ from 1 to $num$ do
 4:         $w_o \leftarrow$ EXPONENTIAL_OBFUSCATION($w_i, W_p, \varepsilon, \alpha_{eucl}, \alpha_{sim}$)
 5:         $w_d \leftarrow$ EXP_ATTACK($w_o, W_p, table$)
 6:         Parse $(\cdot, \cdot, trace_i) \leftarrow w_i$
 7:         Parse $(\cdot, \cdot, trace_d) \leftarrow w_d$
 8:         $d_{sim} \leftarrow$ COMPUTE_SIMILARITY($trace_i, trace_d$)
 9:         Append $d_{sim}$ to $arr\_d_{sim}$
10:     end for
11:     return $arr\_d_{sim}$
12: end function
```



(a) KRR



(b) Exponential Mechanism with (α_{eucl} = 0.75, α_{sim} = 0.25)

Fig. 19. 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 9) is included as a blue line.

whose trace is "similar" to the original wallet's trace is not much better than just guessing the original wallet. 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.