Differentially Private Billing with Rebates

George Danezis[1], Markulf Kohlweiss[1], and Alfredo Rial[2]

[1] Microsoft Research
Cambridge, UK
{gdane,markulf}@microsoft.com
[2] ESAT-COSIC / IBBT
KU Leuven, Belgium
alfredo.rial@esat.kuleuven.be

Abstract. A number of established and novel business models are based on fine grained billing, including pay-per-view, mobile messaging, voice calls, pay-as-you-drive insurance, smart metering for utility provision, private computing clouds and hosted services. These models apply fine-grained tariffs dependent on time-of-use or place of-use to readings to compute a bill.
We extend previously proposed billing protocols to strengthen their privacy in two key ways. First, we study the monetary amount a customer should add to their bill in order to provably hide their activities, within the differential privacy framework. Second, we propose a cryptographic protocol for oblivious billing that ensures any additional expenditure, aimed at protecting privacy, can be tracked and reclaimed in the future, thus minimising its cost. Our proposals can be used together or separately and are backed by provable guarantees of security.

1 Introduction

A number of business models are based on billing customers for fine grained use of a system or resource: mobile network providers charge per call length and type, pay-per-view TV providers charge for the actual requested content. Newer businesses rely heavily on fine grained recordings of activity for billing. Pay-as-you-drive automotive insurance bills drivers per mile depending on the type of road and time of travel. Electronic tolling and congestion charging schemes have been proposed on similar lines. Smart-metering for electricity and gas is being rolled out in the EU and the US in the next few years. Finally, private cloud provision as well as hosted on-line service provision might rely on fine-grained measurements of CPU time usage, memory allocation, disk storage, peak bandwidth, or even the demand and network congestion at the time of day.

The downside of fine-grained metering and billing is the potential threat to privacy. A common privacy-invasive architecture to support such billing consists of providers collecting all usage information in order to apply the appropriate tariffs. Privacy-friendly protocols have also been developed: it is possible to cryptographically combine certified readings with a tariff policy to produce a certified bill that leaks no additional information about the detailed readings [1, 2]. Yet, even the final bill, which is for instance aggregated over a period of usage, may leak information or be used to leak specific readings.

This work makes two contributions to the field of privacy-friendly metering and billing. First, we discuss how to eliminate incidental, accidental or deliberate leakages of information resulting from disclosing the final bill. We show that by adding some, in the long run small, amount of noise it is possible to offer strong privacy guarantees on what an adversary can infer from the final bill. This problem is similar to covert channel minimization [3], and we use techniques from differential privacy that could be more widely applicable. Second, we attempt to minimise the cost of privacy through a cryptographic oblivious billing mechanism. The true cost of service provision is tracked across billing periods, but not revealed to the service provider, which can only verify the deposited funds cover costs. This allows customers to determine the levels of privacy they require and even get a rebate for the additional funds they used to protect their privacy.

Throughout this work we motivate our protocols through the example of a leased private computation cloud. A service provider installs a cloud of 10000 CPUs in the premisses of a large government intelligence agency. In our example, billing is performed on the basis of the compute hours actually used at a fixed rate of $\$0.12$ per CPU instance / hour[3]. A more complex tariff scheme where each hour in the year is costed differently is also supported. The government agency needs to settle the bill each month, but is worried that the amount of computation on particular days is leaked to its adversaries. We will show how our protocols can be used to reduce any leakage below a desired level.

Discussion of the state-of-the-art. Deployed systems for fine grained billing usually employ procedural access control mechanisms to protect privacy: usage data is gathered, and often stored centrally for the purposes of billing. Access control allows only designated parties and processes to access the data, and encryption technology might be used to protect storage and communications. Despite those protections, the fact that personal information is under the control of a service provider raises privacy concerns. A pilot deployment of a pay-as-you-drive insurance scheme by Norwich Union failed, stating privacy concerns as a leading reason for low uptake.[4]

Two types of privacy preserving metering and billing have been proposed in the literature. First, a meter can be entrusted with applying a fine grained tariff to the usage data and only communicating to the service provider a final total fee. In this setting the meter has to be trusted by the users and the service providers both for privacy and correctness. This is usually achieved through trusted hardware and certification. In the automotive setting, where meters record positions of cars for tolling, spot checks have also been proposed to verify the correctness of the meter operation [1]. The second architecture requires meters to cryptographically certify readings and securely hand them over to a user device or service. Cryptographic operations can then be used to apply a tariff scheme, and output a bill along with the necessary cryptographic proofs that certify its correctness. Meters are simpler, and any device can be used to compute bills [2].

[3] The value of a standard compute instance / hour on Amazon EC2 and Microsoft Azure in December 2010.

[4] Insurer stops ‘pay as you drive’, BBC Radio 4’s Money Box `http://news.bbc.co.uk/2/hi/programmes/moneybox/7453546.stm`

Both architectures achieve the same goal: the bill and other necessary information are made available to the service provider, but further information on detailed readings is hidden from it and only available to the consumer.

In this work we are concerned with the remaining information leakage from privacy-preserving billing systems. The value revealed by the protocols, namely the value of the bill, could leak information or be used as a covert channel.

To illustrate the threat, consider a resource consumed in a number of i_{max} distinct time periods i, for $i \in [0, i_{max}]$. Some consumption takes place at each time period i denoted by $c_i \in [0, c_{max}]$, that should be billed at a tariff of p_i per unit. Thus the final bill for all periods should be $B = \sum_{i=0}^{i_{max}} c_i \cdot p_i$. Without making any assumptions on the consumption patterns, as they are out of the system designer's control, it is difficult to estimate what information may be leaking from the final value B. For example an adversary may know, through some side information, that the user consumed only in a single time period T. In such a case the exact value of c_T can be inferred straightforwardly by computing $c_T = B/p_T$. This example threat illustrates that a solution to this problem should make no assumptions about the consumption pattern, assume that arbitrary side-information is available to the adversary, and work for arbitrary (but known) tariff schemes.

We will use a trivial solution as a benchmark to evaluate our own proposals: the user could always pay an amount equivalent to the maximum possible consumption. In the example used so far, this would be: $\max B = c_{max} \cdot \sum_{i=0}^{i_{max}} p_i$. While this is an adequate solution from a privacy perspective, it nullifies the benefits of fine-grained billing as users end up paying a fixed premium irrespective of their consumption. Furthermore it is very wasteful, if the objective is to hide usage of the private cluster at the granularity of an hour or a day.

Outline. Our techniques provide guarantees of privacy depending on the level of protection required by the customers, as well as a cryptographic scheme to amortise the cost of such privacy provision. In Section 2 we study how much noise one needs to add to a bill to ensure specific consumption windows are protected. In Section 4 we propose a cryptographic rebate protocol that keeps a hidden track of the actual amounts due accross multiple billing periods, allowing users to reclaim some of the extra payments made. The rebate protocols also support deposits, anonymous payments using e-cash, and negative bill noise, and prevent abuse by ensuring the funds paid cover the costs of consumption.

2 Differential Privacy for Billing

We start from the premise that customers can add some "noise" to their bill in order to hide their exact usage at specific times. Of course this billing noise represents real money, so they wish to minimise it for a given level of protection required. The first problem we tackle is to determine how much more a customer should pay to hide their pattern of activity for a particular time frame.

Differential privacy was developed as a framework for hiding personal records within databases [4]. A statistic extracted from a database is differentially private if it is nearly as likely as if it was extracted from a database with an arbitrary record removed. This definition encapsulates the intuition that a single individual's record does not overwhelmingly affect the statistic in a way that information about the record might leak.

We have to modify this definition as well as its precise mathematical counterpart to make it applicable to the billing setting. We consider as our database the set of all readings from a meter. In the case of billing private cloud usage each record represents the number of CPUs used for each hour of the billing period. The customer then has to specify its privacy goal: for example they may wish to hide their activity at any arbitrary hour or any arbitrary day of computing. Then they should determine the quality of the protection provided, in terms of how much information the bill reveals about any particular period. Using those parameters we can calculate the additional amount to bill in order to achieve the desired privacy goals.

2.1 Privacy definitions

For simplicity we consider fixed size databases corresponding to a fixed term billing period. For our application this is sufficient, as we are primarily interested in the number of CPU instances used during each hour of the pricing period. For this reason the domain of all possible data sets is described as the Cartesian product: $\mathcal{D} = \{0, \ldots, c_{max}\}^{i_{max}}$. For our private cloud scenario c_{max} is the number of instances in the private cloud, and i_{max} is the number of records per billing period. In our concrete example $c_{max} = 10000$ and i_{max} is the number of hours in a month or a year.

First we define the "distance" between two sets of readings, and repeat some key definitions and results from differential privacy [4], upon which we will be building.

Definition 1. *The record distance $RDist(D_1, D_2)$ between two data sets $D_1, D_2 \in \mathcal{D}$ corresponds to the number of elements (records) in which D_1 and D_2 differ.*

Definition 2. *A randomized function K gives ϵ-differential privacy if for all data sets $D_1, D_2 \in \mathcal{D}$ with $RDist(D_1, D_2) \leq 1$, and all $S \in \Sigma_{Image(K)}$,*[5]

$$Pr[K(D) \in S|D = D_1] \leq \exp(\epsilon) \times Pr[K(D) \in S|D = D_2] \ .$$

The probability is taken over the randomness of K.

Intuitively, mechanisms fulfilling this definition address concerns that an individual might have about filling in one record truthfully, rather than arbitrarily. Differential privacy guarantees that no output (and thus consequences of outputs) becomes significantly more or less likely. In our case the randomized function K will be the billing amount increased by some random value.

A further observation about hiding multiple records k from a database will also prove useful:

[5] A σ-algebra over a set X is a set $\Sigma_X \subset 2^X$ such that $\emptyset \in \Sigma_X$; $S \in \Sigma_X \Rightarrow (X \setminus S) \in \Sigma_X$; and for any $(S_i)_{i \in \mathbb{N}}$, $S_i \in \Sigma_X$, $\bigcap S_i \in \Sigma_X$.

Definition 3. *A randomized function K gives (k, ϵ)-differential privacy if for all data sets $D_1, D_2 \in \mathcal{D}$ with $RDist(D_1, D_2) \leq k$, and all $S \in \Sigma_{Image(K)}$,*

$$Pr[K(D) \in S | D = D_1] \leq \exp(\epsilon \cdot k) \times Pr[K(D) \in S | D = D_2] \ .$$

The probability is taken over the randomness of K.

Lemma 1. *A ϵ-differentially private privacy mechanism K is also (k, ϵ)-differentially private.*

Lemma 1 follows from Definition 3, and shows that the same privacy mechanism K can obstruct inferences on multiple records. In such cases it provides a lower amount of privacy (i.e. $\epsilon' = \epsilon \cdot k$). Hence if a mechanism is to be used to protect multiple records suitable security margins should be provided.

Differentially private mechanisms. The classical differential privacy mechanism by Dwork [4] adds Laplacian noise to the outcome of a query, parametrised by the "sensitivity" of the function f.

Definition 4. *The sensitivity of a function $f : \mathcal{D} \to R^n$ is the maximum distance between output values for which the domain differs in at most one record:*

$$\Delta_f = \max_{\substack{D_1, D_2 \in \mathcal{D} \\ RDist(D_1, D_2) \leq 1}} \|f(D_1) - f(D_2)\|_1$$

For $n = 1$ the sensitivity of f is the maximum difference $|f(D_1) - f(D_2)|$ between pairs of databases D_1, D_2 that differ in only one element. It is shown in [4] that if $f : \mathcal{D} \to R$ is a function with sensitivity Δ_f, then $K(D) = Lap(f(D), \Delta_f/\epsilon)$ is differentially private.

Our adaptations of the differential privacy definitions. Instead of bounding the ratio between output probabilities of actual vs. arbitrary information for a single hourly record, we want to give customers the option of hiding an arbitrary period of time. For example we may want to hide specifics of daily (chunks of 24 records) or weekly (chunks of 168 records) consumption. We call the period length a user is concerned with the *privacy unit*. Furthermore we need to achieve this for statistics in discrete domains (not continuous function), that can only make the bills bigger, never smaller.

Definition 5. *The u-distance $Dist_u(D_1, D_2)$, e.g., $u \in \{hourly, dayly, weekly\}$ between two data sets $D_1, D_2 \in \mathcal{D}$ corresponds to the number of u-units (collection of records) in which D_1 and D_2 differ.*

Our pricing scheme maps each $D \in \mathcal{D}$, $D = (c_1, \ldots, c_{i_{max}})$ to a discrete price: $price(D) = \sum_{i=1}^{i_{max}} c_i \cdot p_I$, where i_{max} is the number of records per billing period, and p_I is the price per hour per instance. Rather than having continuous positive and negative noise as in the original Laplacian differential privacy mechanism, we want to only add discrete positive noise.

If we consider only privacy mechanisms with discrete outputs, we can simplify the differential privacy definition. For discrete distributions, $\Sigma_{Image(K)} = 2^{Image(K)}$, and

$Pr[K(D) \in S] = \sum_{r \in S} Pr[K(D) = r]$. Definition 8 can thus be restated as the following equation: $\sum_{r \in S} Pr[K(D) = r | D = D_1] \leq \exp(\epsilon) \cdot \sum_{r \in S} Pr[K(D) = r | D = D_2]$. From this we derive an alternative definition for differential privacy for discrete distributions:

Definition 6. *A randomized function K gives ϵ-differential u-privacy if for all data sets $D_1, D_2 \in \mathcal{D}$ with $Dist_u(D_1, D_2) \leq 1$, and all $r \in Image(K)$,*

$$Pr[K(D) = r | D = D_1] \leq \exp(\epsilon) \times Pr[K(D) = r | D = D_2] \ .$$

The probability is taken over the randomness of K.

Lemma 2. *Definition 8, Definition 3, and Lemma 1 apply to u-privacy:*

1. *For discrete privacy mechanisms Definition 8 and Definition 6 for $u = hourly$ are equivalent.*
2. *Let n_u be the number of records in a u-unit. If K is (n_u, ϵ)-differential hourly-private, then K is also $(n_u \cdot \epsilon)$-differential u-private.*

Dwork [5] notes that, because of the multiplicative nature of the definition, an output whose probability is zero on a given database must also have probability zero on any neighboring database, and therefore, by repeated application of the definition, on any other database.

Handling privacy mechanisms that result in distributions for which the support of $K(D_1)$ and $K(D_2)$ may differ requires extra care. Such a situation arises, e.g., when K adds only positive noise. If for instance $price(D_1) < price(D_2)$ to which K adds positive noise. Let ν_{min} be the minimum amount of noise that is added, then the value $r = price(D_1) + \nu_{min}$ is in the support of $K(D_1)$ but has 0 probability for $K(D_2)$. It follows that such a mechanism can never be differentially private.

To overcome this problem, we define partial differential privacy. A statistic offers partially differential u-privacy if it is differentially private for all outputs in the overlapping support of any two databases D_1 and D_2 with $Dist_u(D_1, D_2) \leq 1$. Furthermore we require the probability that the output of the statistic is not in the overlapping domains to be bound by a small probability δ. This means that the function is differentially private most of the time (or with probability at least $1 - \delta$).

Definition 7. *A randomized function K gives δ-partially ϵ-differential u-privacy if the following two properties hold:*

1. *For all $D_1, D_2 \in \mathcal{D}$ with $Dist_u(D_1, D_2) \leq 1$, and all $r \in Supp(K(D_1)) \cap Supp(K(D_2))$,*

$$Pr[K(D_1) = r] \leq \exp(\epsilon) \times Pr[K(D_2) = r] \ .$$

2. *For all data sets $D_1, D_2 \in \mathcal{D}$ with $Dist_u(D_1, D_2) \leq 1$,*

$$Pr[r \leftarrow K(D_1) : r \notin Supp(K(D_2))] < \delta \ .$$

For both properties, the probability is taken over the randomness of K.

As for the traditional differential privacy definitions, longer periods of privacy can be guaranteed with lower security parameters:

Lemma 3. *Let n_u be the number of records in a u-unit. If K is δ-partially (n_u, ϵ)-differential hourly-private, then K is also $(n_u \cdot \delta)$-partially $(n_u \cdot \epsilon)$-differential u-private*

Proof. Consider the joint distribution of K for all D_1 and D_2 with $RDist(D_1, D_2) \leq n_u$. The probability of drawing a value r not in the domain of at least one of $K(D_i)$ is $\delta' \leq 1-(1-\delta)^{n_u} \leq n_u \cdot \delta$. This proves Property 2 for partial differential privacy. If r is in the domain, Property 1 is proved as in Lemma 2. □

Given the above definition for privacy we propose a concrete mechanism to obscure readings. We simply add to the bill $f(D)$ for consumption D an amount of noise drawn from a Geometric distribution with parameter $p = \epsilon/\Delta_{f,u}$.[6] The sensitivity $\Delta_{f,u}$ is the maximum difference of a bill between two databases D_1 and D_2 differing in at most 1 u-unit (e.g. an hour, a day, or a week). Similarly, ϵ is a security parameter expressing information leakage.

Claim. Let $f : \mathcal{D} \to R$ be a function with sensitivity $\Delta_{f,u}$, then $K(D) = f(D) + Geo(\epsilon/\Delta_{f,u})$ is $(2 \cdot \epsilon)$-partially ϵ-differentially u-private.

Proof. We prove Property 1 as follows: assume $K(D_1)$ outputs r that is also in the domain of $K(D_2)$. For $p = \epsilon/\Delta_{f,u}$ and making use of the bound $(1+\alpha)^k \leq e^{\alpha k}$:

$$\frac{\Pr[K(D_1) = t|D_1]}{\Pr[K(D_2) = t|D_2]} = \frac{(1-p)^{(t-f(D_1))}p}{(1-p)^{(t-f(D_2))}p} = (1-\epsilon/\Delta_{f,u})^{(f(D_2)-f(D_1))}$$
$$\leq \left(1 - \frac{\epsilon}{(f(D_1)-f(D_2))}\right)^{(f(D_2)-f(D_1))} \leq e^{\epsilon}$$

We prove Property 2 as follows: we show that the probability that r is not in the domain of $D(D_2)$ is bound by $\delta = 2\epsilon$:

$$Pr[r \leftarrow K(D_1) : r \notin Supp(K(D_2))] =$$
$$\Pr[Geo(\epsilon/\Delta_{f,u}) < \Delta_{f,u}] = 1 - (1 - \frac{\epsilon}{\Delta_{f,u}})^{\Delta_{f,u}+1} \leq 2\epsilon \ .$$

□

As also noted by [7], the application of a public function on the outputs of a differentially private statistic does not leak any additional information. We can modify the billing function to only charge up to the maximum possible consumption: $K'(D) = \min(f(D) + Geo(\epsilon/\Delta_{f,u}), \max_{D'} f(D'))$. Intuitively we use geometric noise, as this adds the maximal uncertainty for a given mean. The variant of the geometric distribution with support for negative and positive integers defined as $\Pr[k] = \frac{1}{2}(1-p)^{|k|}p$ is the discrete equivalent of the Laplace distribution, and would also provide differentially private guarantees. We limit ourselves to the proposed noise distribution to ensure users only add positive noise to their bills.

[6] Two-sided Geometric noise was also proposed in [6] as a differential privacy mechanism.

Interpretation of differential privacy in terms of inference. From the attackers perspective the goal of collecting statistics about the output of the privacy mechanism K is to infer something about the underlying database. For instance, the attacker might want to distinguish between two databases D_1 and D_2, in the sense of semantic security.

Differential privacy does not guarantee anything about the probability ratio (likelihood ratio) between databases D_1 and D_2 with $Dist_u(D_1, D_2) \leq 1$ given an observed outcome of K; it merely says that this ratio will differ only by a small factor from the ratio of the prior. Note that because D_1 and D_2 are interchangeable, the new ratio is also bounded from below.

Lemma 4. *Given an observed outcome of a differentially private K the probability ratio (likelihood ratio) between databases D_1 and D_2 with $Dist_u(D_1, D_2) \leq 1$ differs by less than a factor* $\exp(\epsilon)$ *from the ratio of the prior.*

$$\frac{Pr[D = D_1|K(D) = r]}{Pr[D = D_2|K(D) = r]} \leq \exp{(\epsilon)} \times \frac{Pr[D = D_1]}{Pr[D = D_2]}.$$

Proof. From Bayes theorem we can write:

$$Pr[D = D_i|K(D) = r] = \frac{Pr[K(D) = r|D = D_i] \times Pr[D = D_i]}{Pr[K(D) = r]}$$

whence, since K is differentially private, we can write:

$$\begin{aligned}\frac{Pr[D = D_1|K(D) = r]}{Pr[D = D_2|K(D) = r]} &= \frac{Pr[K(D) = r|D = D_1]}{Pr[K(D) = r|D = D_2]} \times \frac{Pr[D = D_1]}{Pr[D = D_2]} \\ &\leq \exp{(\epsilon)} \times \frac{Pr[D = D_1]}{Pr[D = D_2]}\ .\end{aligned}$$

□

The cost of privacy. Obscuring bills by adding noise may lead to paying extra for a service. Customers have incentives to minimise their costs for a desired level of privacy protection. We provide a few illustrative examples of the average extra cost involved in settling a bill for different privacy units of an hour, a day or a week. In our usual example we consider a private cloud of 10K CPUs, billed as \$0.12 a CPU / hour. We denote as $\beta = f(D)$ the actual service cost associated with the use of the service for a year.

It is clear from Table 1 that providing a differentially private bill for more than a single hourly period is an expensive business. The proposed mechanism allows for lower overheads for yearly bills when customers wish to protect arbitrary hours or days in the year. When it comes to protecting arbitrary weeks this protection is only offered with a low security parameter ($\epsilon = 0.1$). Why is the cost so high? It is because the privacy guarantee offered is very strong: no matter what side information the adversary has, including the detailed readings for other periods, they should not be able to infer information about an arbitrary privacy unit. For example if the adversary knows the exact consumption for the other 364 days they should still not learn more than permitted about the last day. This is a very strong guarantee and as a result it comes at a high cost, when applied directly.

Privacy units	Security (ϵ)	Pay Monthly	Pay Yearly	Fixed Rate
Hourly	0.1	$\beta + \$144,000$	$\beta + \$12,000$	$\$10,512,000$
(units = 1)	0.01	$\beta + \$1,440,000$	$\beta + \$120,000$	$\$10,512,000$
Daily	0.1	$\beta + \$3,456,000$	$\beta + \$288,000$	$\$10,512,000$
(units = 24)	0.01	$(\$10,512,000)$	$\beta + \$2,880,000$	$\$10,512,000$
Weekly	0.1	$(\$10,512,000)$	$\beta + \$2,016,000$	$\$10,512,000$
(units = 168)	0.01	$(\$10,512,000)$	$(\$10,512,000)$	$\$10,512,000$

Table 1. Yearly average bill after the application of the privacy mechanism K' compared with the fixed rate privacy mechanism. Different values of the security parameter (ϵ), different privacy units (hourly, daily and weekly) as well as the options of paying monthly or yearly are presented. Amounts in parenthesis indicate that the expected cost is higher than paying for the maximum consumption.

Table 1 also contains the cost of paying bills monthly, which incur a 12 fold overhead for the same level of protection. It is clear that there are advantages in paying in batches if in fact the desired property is to hide any fixed period of time within the billing period (an hour, a day, a week). We will see in the next section how we can do better than this: we can aggregate the true cost of service provision, and use cryptographic methods to reclaim most of the additional cost of privacy in the long term without sacrificing any security.

Longer guarantees. Degradation of privacy in our framework is graceful, since some privacy guarantees are provided for periods longer than what is strictly defined by the chosen u-units. For example a user may choose a partially ϵ-differential function $K_{\epsilon,24}$ providing u-privacy for a day (i.e. 1 u-unit = 24 hourly periods) with $\epsilon = 0.01$. In our standing example this means he should add an extra amount to his bill drawn from a Geometric distribution with parameter $\$2,880,000$. What does that guarantee? Let's assume the adversary knows the exact consumption about all days except for one. Furthermore the adversary knows that the consumption on the target day could only have taken one out of two values with equal probability: this means that the ratio of priors $\frac{Pr[D=D_1]}{Pr[D=D_2]} = 1$. Then after receiving information about the bill the adversary would at best know that $0.99 \approx 1/(1+\epsilon) \approx 1/e^{\epsilon} \leq \frac{Pr[D=D_1|K'(D)]}{Pr[D=D_2|K'(D)]} \leq e^{\epsilon} \approx 1+\epsilon = 1.01$. This is a small amount of information.

Now let's consider an adversary that tries to infer something over a longer period, e.g., a week. The adversary knows all user consumption outside this target week, and furthermore knows that user consumption within the week could only have been one of two possibilities D'_1 or D'_2 with equal probability as before. Due to Lemma 3 we know that the $K'(D)$ scheme is also partially ϵ-differentially private for a longer u-unit of a week (1 weakly-unit = 7×24 hourly-units), with a new security parameter $\epsilon' = \epsilon \cdot 7$. This means that the new posterior ratio of probabilities over the two only possible outcomes is $0.93 \approx 1/(1+\epsilon') \approx 1/e^{\epsilon'} \leq \frac{Pr[D=D_1|K'(D)]}{Pr[D=D_2|K'(D)]} \leq e^{\epsilon'} \approx 1+\epsilon' = 1.07$. Despite the lower degree of privacy, some quantifiable protection is still available against longer-term profiling.

Limitations. Our variant of differential privacy relies on only introducing positive noise. This is desirable as it guarantees that the bill at least covers the cost of service provision. At the same time this provides a one sided security property: a final bill can always be confused with a lower bill, but not always with a higher bill. For example there is a positive probability that a sensitive day passes with no consumption and then no noise is added to the bill. If an adversary knows all other consumptions in the year, they can infer that indeed no consumption took place on the unknown day. Our mechanism thus assumes that the baseline of no consumption is not as sensitive as high consumption.

While information leakage about low levels of consumption is possible, it is not very likely for high levels of security as characterised by the security parameter δ.

Summary. We have shown that adding noise to the bill can provide high levels of security parametrised by a parameter ϵ and a privacy unit. This security holds even against adversaries with knowledge of many readings. At the same time this comes with a high overhead. In the next section we show that the bulk of the cost of providing privacy can be recuperated in the long run. We achieve this by keeping hidden accounts of what is actually due for service provision, versus what has been paid. In the long run users can only add the necessary noise to keep their accounts positive, including negative noise – while ensuring that their funds cover their consumption.

3 Cryptographic Building Blocks of the PBR Protocol

Signature Schemes A signature scheme consists of the algorithms (Keygen,Sign,Verify). Keygen(1^k) outputs a key pair (sk, pk). Sign(sk, m) outputs a signature s on message m. Verify(pk, s, m) outputs accept if s is a valid signature on m and reject otherwise. This definition can be extended to support multi-block messages $\boldsymbol{m} = \{m_1, \ldots, m_n\}$. Existential unforgeability [8] requires that no p.p.t. adversary should be able to output a message-signature pair (s, m) unless he has previously obtained a signature on m.

Commitment schemes A non-interactive commitment scheme consists of the algorithms ComSetup, Commit and Open. ComSetup(1^k) generates the parameters of the commitment scheme par_c. Commit(par_c, x) outputs a commitment c_x to x and auxiliary information $open_x$. A commitment is opened by revealing $(x, open_x)$ and checking whether Open$(par_c, c_x, x, open_x)$ outputs accept. A commitment scheme has a hiding property and a binding property. Informally speaking, the hiding property ensures that a commitment c_x to x does not reveal any information about x, whereas the binding property ensures that c_x cannot be opened to another value x'.

Our protocols make heavy use of homomorphic commitment schemes. A commitment scheme is said to be additively homomorphic if, given two commitments c_{x_1} and c_{x_2} with openings $(x_1, open_{x_1})$ and $(x_2, open_{x_2})$ respectively, there exists an operation $\otimes$ such that, if $c = c_{x_1} \otimes c_{x_2}$, then Open$(par_c, c, x_1 + x_2, open_{x_1} + open_{x_2})$ outputs accept.

Proofs of Knowledge A zero-knowledge proof of knowledge [9] is a two-party protocol between a prover and a verifier. The prover convinces the verifier who knows only a public proof *instance* that he knows a secret input (called *witness*) that allows him to prove that the public and the secret value together fulfill some relational statement $(witness, instance) \in R$ without disclosing the secret input to the verifier. The protocol should fulfill two properties. First, it should be a proof of knowledge, i.e., a prover without knowledge of the secret input convinces the verifier with negligible probability. More technically, there exists a knowledge extractor that extracts the secret input from a successful prover with all but negligible probability. Second, it should be zero-knowledge, i.e., the verifier learns nothing but the truth of the statement. More technically, for all possible verifiers there exists a simulator that, without knowledge of the secret input, yields a distribution that cannot be distinguished from the interaction with a real prover. Witness indistinguishability is a weaker property that requires that the proof does not reveal which witness (among all possible witnesses) was used by the prover.

We use several existing results to prove statements about discrete logarithms: proof of knowledge of a discrete logarithm [10]; proof of knowledge of the equality of elements in different representations [11]; proof with interval checks [12], range proof [13] and proof of the disjunction or conjunction of any two of the previous [14]. These results are often given in the form of Σ-protocols but they can be turned into non-interactive zero-knowledge arguments in the random oracle model via the Fiat-Shamir heuristic [15].

When referring to the proofs above, we follow the notation introduced by Camenisch and Stadler [16] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. $\mathsf{NIPK}\{(\alpha, \beta, \delta) : y = {g_0}^{\alpha} {g_1}^{\beta} \wedge \tilde{y} = \tilde{g_0}^{\alpha} \tilde{g_1}^{\delta} \wedge A \leq \alpha \leq B\}$ denotes a "*zero-knowledge Proof of Knowledge of integers α, β, and δ such that $y = {g_0}^{\alpha} {g_1}^{\beta}$, $\tilde{y} = \tilde{g_0}^{\alpha} \tilde{g_1}^{\delta}$ and $A \leq \alpha \leq B$ holds*", where $y, g_0, g_1, \tilde{y}, \tilde{g_0}, \tilde{g_1}$ are elements of some groups $G = \langle g_0 \rangle = \langle g_1 \rangle$ and $\tilde{G} = \langle \tilde{g_0} \rangle = \langle \tilde{g_1} \rangle$ that have the same order. (Note that some elements in the representation of y and $\tilde{y}$ are equal.) The convention is that letters in the parenthesis, in this example α, β, and δ, denote quantities whose knowledge is being proven, while all other values are known to the verifier. We denote a non-interactive proof of signature possession as $\mathsf{NIPK}\{(x, s_x) : \mathsf{Verify}(pk, s_x, x) = \text{accept}\}$.

4 Private Billing with Rebates

We have seen that one way of protecting privacy involves adding 'noise' to the bill to be payed for a certain period. Yet, the amount of noise can become significant particularly to achieve a high quality of privacy or privacy for longer periods within the billing time frame. For this reason we develop a complementary oblivious billing protocol that can be used to alleviate those concerns. Its key features include:

- The ability to maintain a hidden bill of actual consumption that can be used to reclaim any excess used for protecting privacy at a later time.
- A mechanism for proving that the amount payed to the utility provider exceeds the bill for actual consumption without revealing the actual bill.
- Support for an initial deposit to support later use of positive as well as negative noise for the bills.

– Compatibility with anonymous e-cash schemes allowing bills to be settled anonymously, as well as advanced privacy friendly payment mechanisms that allow users to hide the amounts actually payed to the utilities.

We discuss in detail and prove the correctness of the billing protocols, and the mechanisms to ensure payments exceed the amount consumed. The specifics of optional e-cash protocols that allow hidden payments are beyond the scope of this work, and we leave their detailed description to future work.

Our oblivious payment protocols can be used to reclaim in the long run an excess payed as a result of a differentially private billing mechanism as presented in the previous sections. With the deposit facility, adding negative noise is possible, as long as the overall balance of payments stays positive. The protocols can also be used to support the naive mechanism where a bill for maximal consumption is payed, and allow parties to later reclaim some of it back. Finally given anonymous e-cash they can be used to provide full oblivious payments without the need to add any noise to the bills, as they never need to be revealed (technically: $noise = -fee$). Which variant to use therefore depends on the infrastructure available and the degree of complexity parties are ready to accept.

4.1 The PSM protocol

We will be building upon PSM (Privacy-Preserving Smart Metering), a cryptographic protocol for privacy-friendly billing [2]. PSM mediates interactions among three parties: a meter M that outputs consumption data $cons$ and related information $other$; a service provider P that establishes a pricing policy Υ and a user U that receives consumption readings from meter M and at each billing period pays a fee to provider P. The pricing policy Υ is a public function that takes consumption data $cons$ together with other information $other$ (e.g., the time of consumption) and computes a price. The overall price $price(D) = \sum_{i=1}^{|D|} price_i$ is computed by adding the prices corresponding to the individual consumptions in a billing period. For our running private cloud example, $\Upsilon(cons, other) = cons \cdot 0.12$ and does not depend on $other$. As in the original protocols we assume a tamper resistant meter is used to provide accurate and appropriately cryptographically packaged readings. These can be processed by the user to prove their bill in zero-knowledge to the provider. At this point users may also choose to add some noise to ensure differential privacy.

The security of PSM is shown in the simulation-based security paradigm [17–19]. In the real world, the protocol $\mathsf{PSM}(\mathrm{M}, \mathrm{P}, \mathrm{U})$ is run in an adversarial environment that may corrupt some of the protocol parties, indicated by $\tilde{\mathrm{M}}$, $\tilde{\mathrm{P}}$, $\tilde{\mathrm{U}}$. Corrupted parties just forward messages between the environment and honest protocol participants. In the ideal world, dummy protocol parties D_{M}, D_{P}, D_{U} run an ideal protocol $\mathsf{Ideal}(\mathcal{F}_{\mathsf{PSM}}, D_{\mathrm{M}}, D_{\mathrm{P}}, D_{\mathrm{U}})$ by just forwarding messages to an ideal functionality $\mathcal{F}_{\mathsf{PSM}}$. Uncorrupted $D_x, x \in \{\mathrm{M}, \mathrm{P}, \mathrm{U}\}$ interact with the environment while corrupted dummy parties $\tilde{D}_x$ interact with a simulator Sim.

We consider w.l.o.g. a corrupted provider $\tilde{\mathrm{P}}$ and say that a protocol is secure against $\tilde{\mathrm{P}}$, if there exists a simulator Sim such that no environment Env can tell whether it is interacting with $\mathsf{PSM}(\mathrm{M}, \tilde{\mathrm{P}}, \mathrm{U})$ or with $\mathsf{Sim} \| \mathsf{Ideal}(\mathcal{F}_{\mathsf{PSM}}, D_{\mathrm{M}}, \tilde{D}_{\mathrm{P}}, D_{\mathrm{U}})$. Conceptually

Sim translates influence that Env has through $\tilde{\mathrm{P}}$ on the protocol into influence on $\mathcal{F}_{\mathsf{PSM}}$ through $\tilde{D}_{\mathrm{P}}$, and leakage that $\tilde{D}_{\mathrm{P}}$ receives from $\mathcal{F}_{\mathsf{PSM}}$ into leakage that Env could learn from $\tilde{\mathrm{P}}$. Similarly, PSM is proven secure against a corrupted user $\tilde{\mathrm{U}}$.

Listing 1 Functionality $\mathcal{F}_{\mathsf{PBR}}$

$\mathcal{F}_{\mathsf{PBR}}$ is parameterized by deposit relation R and a policy set Y and interacts with dummy parties D_M, D_P and D_U. Initially $T = \emptyset$, $d = 0$, $account = 0$.

On $(\texttt{Policy}, \Upsilon)$ from D_P where $\Upsilon \in Y$
- store Υ; send $(\texttt{Policy}, \Upsilon)$ to D_U

On $(\texttt{Consume}, cons, other)$ from D_M
- increment counter d; add $(d, cons, other)$ to T; send $(\texttt{Consume}, cons, other)$ to D_U

On $(\texttt{Deposit}, (inc, wit), instance)$ from D_U where $balance + inc \geq 0$
- if $((inc, wit), instance) \in R$, let $balance$ += inc, send $(\texttt{Deposit}, instance)$ to D_P

On $(\texttt{Payment}, from, until, noise)$ from D_U where $0 \leq from \leq until \leq d$ and $balance + noise \geq 0$
- for $i = from$ to $until$, calculate $price_i = \Upsilon(cons_i, other_i)$
- let $fee = \sum_{i=from}^{until} price_i + noise$ and $balance$ += $noise$
- send $(\texttt{Pay}, from, until, fee)$ to D_P

4.2 Rebate Ideal Functionality

We propose a new ideal functionality $\mathcal{F}_{\mathsf{PBR}}$ (see Listing 1) that extends the functionality $\mathcal{F}_{\mathsf{PSM}}$. The functionality keeps track of the user's consumptions in a set T containing tuples $(i, cons, other)$. During a payment, the policy Υ is applied to all $(cons, other)$ in the interval $[from, until]$ to compute the price $price_i = \Upsilon(cons_i, other_i)$ per consumption. The overall fee that the user has to pay is computed as $fee = \sum_{i=from}^{until} price_i + noise$. The value $noise$ is added to the fee to improve the user's privacy. The ideal functionality also maintains a balance that corresponds to the sum of all the $noise$ added to payments. Note that the user can get rebates by using negative noise, but that the balance is never allowed to be negative.

The ideal functionality also allows to increase the balance through a deposit. The user has to provide input $((inc, wit), instance) \in R$. The parameterization by relation R allows to support both standard deposit mechanisms that reveal the deposited amount inc as well as advanced deposit mechanisms that hide this value from the provider. In the simple mechanism the user reveals how much he wants to deposit: $wit = \epsilon$ and R corresponds to simple equality, i.e. $R = \{(inc, \epsilon), inc) | inc \in \mathbb{Z}\}$.

To obtain a more advanced privacy-friendly deposit mechanism, the witness could correspond to a one-show anonymous credential $cred$. The relation requires that $cred$ is a one-show credential with an increment value inc and serial number s issued under public key pk_B, i.e, $R = \{((inc, cred), (s, pk_B)) | \mathsf{Verify}(pk_B, cred, (inc, s)) = \mathsf{accept}\}$. The real protocol cryptographically enforces this using a zero-knowledge proof.[7] To obtain such a one-show credential without revealing the value of inc to the

[7] A zero-knowledge proof of knowledge [9] is a two-party protocol between a prover and a verifier. The prover convinces the verifier, who knows only a public proof $instance$, that he knows a secret input (called $witness$) that allows him to prove that the public and the secret value together fulfill some relational statement $(witness, instance) \in R$ without disclosing the secret input to the verifier.

provider, additional infrastructure is needed. In particular such a mechanism seems to require some form of anonymous payment, either physical cash or anonymous e-cash. Given such a payment mechanism, the provider's bank, after receiving an anonymous payment of value inc and depositing this amount on the provider's bank account, could blindly issue the signature $\mathsf{Sign}(pk_B, (inc, s))$ using a partially-blind issuing protocol [20]. The issue protocol guarantees that the bank does not learn s, and thus even if the provider and his bank collude they cannot link the issuing of $cred$ to its use.

Listing 2 Protocol PBR(M, P, U)

Parties M, P, U are parameterized by R and Y and interact over secure channels. All participants have registered public keys generated by Mkeygen, Pkeygen, Ukeygen with a key registration authority $\mathcal{F}_{\mathsf{REG}}$ and keep their private keys secret. P also registers commitment parameters par_c.

On $(\texttt{Policy}, \Upsilon)$ from Env
- P runs $\Upsilon_s \leftarrow \mathsf{SignPolicy}(sk_{\mathrm{P}}, \Upsilon)$ and sends Υ_s to U
- Upon receiving Υ_s, U extracts Υ; if $\Upsilon \notin Y$, he aborts
- if $\mathsf{VerifyPolicy}(pk_{\mathrm{P}}, \Upsilon_s) = 1$, U stores Υ_s, and sends $(\texttt{Policy}, \Upsilon)$ to Env

On $(\texttt{Consume}, cons, other)$ from Env
- M increments d_{M}, runs $\mathrm{SC} \leftarrow \mathsf{SignConsumption}(sk_{\mathrm{M}}, par_c, cons, other, d_{\mathrm{M}})$ and sends (SC) to U
- Upon receiving (SC), U runs $b \leftarrow \mathsf{VerifyConsumption}(pk_{\mathrm{M}}, par_c, \mathrm{SC}, d_{\mathrm{U}} + 1)$
- if $b = 1$, U increments d_{U}, adds SC to T_{U}, parses SC as $(d_{\mathrm{M}}, cons, open_{cons}, c_{cons}, other, open_{other}, c_{other}, \mathrm{sc})$, and sends $(\texttt{Consume}, cons, other)$ to Env

On $(\texttt{Deposit}, (inc, wit), instance)$ from Env where
$balance + inc \geq 0$ and $((inc, wit), instance) \in R$
- U runs $(aux', D) \leftarrow \mathsf{Deposit}(par_c, (inc, wit), instance, aux, R)$
- U sets $balance$ += inc and $aux = aux'$ and sends $(D, instance)$ to P
- Upon receiving $(D, instance)$, P runs $(c'_{balance}, b) \leftarrow \mathsf{VerifyDeposit}(par_c, D, c_{balance}, instance, R)$
- if $b = 1$, he sets $c_{balance} = c'_{balance}$ and sends $(\texttt{Deposit}, instance)$ to Env

On $(\texttt{Payment}, from, until, noise)$ from Env where
$0 \leq from \leq until \leq d_{\mathrm{U}}$ and $balance + noise \geq 0$
- U runs $(aux', \mathrm{Q}) \leftarrow \mathsf{Pay}(sk_{\mathrm{U}}, par_c, \Upsilon_s, T_{\mathrm{U}}[from : until], noise, aux)$
- U sets $aux = aux'$ and $balance$ += $noise$; U sends $(\mathrm{Q}, from, until)$ to P
- Upon receiving $(\mathrm{Q}, from, until)$, P runs $(fee, c'_{balance}, b) \leftarrow \mathsf{VerifyPayment}(pk_{\mathrm{M}}, pk_{\mathrm{U}}, pk_{\mathrm{P}}, par_c, \mathrm{Q}, c_{balance}, from, until)$
- if b=1, he sets $c_{balance} = c'_{balance}$ and sends $(\texttt{Pay}, from, until, fee)$ to Env

4.3 Rebate Protocol

We propose a new protocol for privacy-preserving billing with rebates (PBR) (see Listing 2) that extends PSM with a mechanism for adding noise, keeping a hidden balance, and making deposits. Like PSM, our protocol operates in the $\mathcal{F}_{\mathsf{REG}}$ hybrid-model [17] where parties register their public keys at a trusted registration entity. As in the original scheme the user receives signed policies from the utility provider P and signed readings from the meter M. The payment transaction only reveals the overall fee, which now can be subject to additional noise.

We extend this protocol with a novel oblivious rebate system that allows the user to get rebates (in the amount of his noise) in future payments. The rebate is implemented using a homomorphic update c_{noise} to a balance commitment $c_{balance}$ that commits the user to his balance towards the provider but keeps the $balance$ itself secret. Our protocol supports an optional Deposit mechanism that allows the user to add or withdraw funds from the rebate $balance$. Value aux contains the opening for a commitment $c_{balance}$ to $balance$. Through the use of zero-knowledge proofs the provider is guaranteed that the value committed to in $c_{balance}$ is updated correctly and never becomes negative.

The protocol parties P, U, and M interact with each other using algorithms Pkeygen, Ukeygen, Mkeygen (for key generation); SignPolicy, SignConsumption, Deposit, and Pay (for generation of input); and VerifyPolicy, VerifyConsumption, VerifyDeposit, and VerifyPayment (for verification of input). The functionality of the meter as well as SignPolicy, SignConsumption, VerifyPolicy, and VerifyConsumption are unchanged from the original scheme.[8] We describe the new Deposit and VerifyDeposit algorithms and the changes to Pay and VerifyPayment:

Listing 3 Algorithms

- $\mathsf{Deposit}(par_c, (inc, wit), instance, aux, R)$. Parse aux as $(balance, open_{balance}, c_{balance})$. Compute commitment $(c_{inc}, open_{inc}) = \mathsf{Commit}(par_c, inc)$ and a non-interactive proof π_{inc}:[9]

$$\begin{aligned}\pi_{inc} \leftarrow \mathsf{NIPK}\{\ (inc, open_{inc}, wit, balance, open_{balance}) :\ & (c_{balance}, open_{balance}) = \mathsf{Commit}(par_c, balance) \wedge \\ & (c_{inc}, open_{inc}) = \mathsf{Commit}(par_c, inc) \wedge \\ & ((inc, wit), instance) \in R \ \wedge\ balance + inc \geq 0\}\ .\end{aligned}$$

  Let $D = (\pi_{inc}, c_{inc})$ and $aux' = (balance + inc, open_{balance} + open_{inc}, c_{balance} \otimes c_{inc})$. Output (aux', D).
- $\mathsf{VerifyDeposit}(par_c, D, c_{balance}, instance, R)$. Parse D as $(\pi_{balance}, c_{inc})$. Verify π_{inc}. If verification succeeds, set $b = 1$ and $c'_{balance} = c_{balance} \otimes c_{inc}$, otherwise set $b = 0$. Output $(c'_{balance}, b)$.
- $\mathsf{Pay}(sk_{\mathrm{U}}, par_c, \Upsilon_s, T, noise, aux)$. Parse aux as $(balance, open_{balance}, c_{balance})$. Compute commitment $(c_{noise}, open_{noise}) = \mathsf{Commit}(par_c, noise)$ and a non-interactive proof π_{noise}:

$$\begin{aligned}\pi_{noise} \leftarrow \mathsf{NIPK}\{\ (noise, open_{noise}, balance, open_{balance}) :\ & (c_{balance}, open_{balance}) = \mathsf{Commit}(par_c, balance) \wedge \\ & (c_{inc}, open_{inc}) = \mathsf{Commit}(par_c, inc) \wedge \\ & balance + noise \geq 0\}\ .\end{aligned}$$

  Let $aux' = (balance + noise, open_{balance} + open_{noise}, c_{balance} \otimes c_{noise})$.

  The rest of the algorithm follows [2]: For each $(i, cons, open_{cons}, c_{cons}, other, open_{other}, c_{other}, \mathrm{sc}) \in T$ where $from \leq i \leq until$, calculate $price_i = \Upsilon(cons, other)$, commitment $(c_{price_i}, open_{price_i}) = \mathsf{Commit}(par_c, price)$, and a proof π_i that $price_i$ was computed correctly according to the policy and the commitments c_{cons}, c_{other}. The proof π_i depends on the policy Υ and can use auxiliary values in Υ_s, see [2] on how to implement different pricing policies.

  Computing $fee = noise + \sum_{i=from}^{until} price_i$ and $open_{fee} = open_{noise} + \sum_{i=from}^{until} open_{price_i}$ gives an opening to a commitment to fee. Let $\mathrm{Q} = (fee, open_{fee}, c_{noise}, \pi_{balance}, \{\mathrm{sc}_i, i, c_{cons_i}, c_{other_i}, c_{price_i}, \pi_i\}_{i=1}^{N})$. Output (aux', Q).

[8] For the sake of brevity we omit the Reveal mechanism of PSM. It would add little new and could be implemented in a straight forward manner using trapdoor commitments.

[9] If R corresponds to equality, the protocol can be simplified to avoid computing c_{inc}.

- $\mathsf{VerifyPayment}(pk_{\mathrm{M}}, pk_{\mathrm{U}}, pk_{\mathrm{P}}, par_c, \mathrm{Q}, c_{balance}, \mathit{from}, \mathit{until})$. Parse Q as $(\mathit{fee}, \mathit{open}_{\mathit{fee}}, c_{noise}, \pi_{balance}, \{\mathrm{sc}_i, d_i, c_{cons_i}, c_{other_i}, c_{price_i}, \pi_i\}_{i=1}^{N})$. Verify π_{noise}. If verification fails, set $b = 0$. Otherwise set $c'_{balance} = c_{balance} \otimes c_{noise}$ and $b = 1$.

  The rest of the algorithm follows [2]: For $i = \mathit{from}$ to until, run $\mathrm{M}\mathsf{verify}(pk_{\mathrm{M}}, \mathrm{sc}_i, \langle i, c_{cons_i}, c_{other_i}\rangle)$ and verify π_i. Set $b = 0$ if any of the signatures or the proofs is not correct. Add the commitments to the prices $c'_{\mathit{fee}} = c_{noise} \otimes \left(\otimes_{i=1}^{N} c_{price_i}\right)$ and execute $\mathsf{Open}(par_c, c'_{\mathit{fee}}, \mathit{fee}, \mathit{open}_{\mathit{fee}})$. If the output is reject set $b = 0$. Output $(\mathit{fee}, c'_{balance}, b)$.

Theorem 1. *Given the security of its building blocks,* PBR *is secure against a corrupted provider* $\tilde{\mathrm{P}}$ *and a corrupted user* $\tilde{\mathrm{U}}$. (See Appendix with supplemental material.)

Using PBR *for differential privacy.* Even an ideal cryptographic billing mechanism as described by the PSM or PBR ideal functionalities cannot protect the user's privacy against an adversary/environment that already knows enough about the user's behavior – possibly including all consumption or additional random noise – to infer privacy sensitive information from the final fee alone. For our privacy analysis we assume that the environment Env is divided into a part $\mathsf{Env}_{\mathrm{U}}$ that is controlled by the user, and a part $\mathsf{Env}_{\tilde{\mathrm{P}}}$ that is controlled by the adversary and that may have some influence on and knowledge about the user's behavior. In the original PSM protocol all the final fee is only the result of the individual consumptions of $\mathsf{Env}_{\mathrm{U}}$ for which the provider may make inferences or gain side information. The PBR protocol gives $\mathsf{Env}_{\mathrm{U}}$ the possibility to obscure the fee with random noise, which is easier to conceal from $\mathsf{Env}_{\tilde{\mathrm{P}}}$.

5 Proof of PBR Protocol

Let $\equiv$ denote computational indistinguishability. The following two claims have been proven about PSM [2]:

Claim (Security of PSM Against Corrupted Provider). Under the unforgeability of the signature schemes $(\mathrm{Mkeygen}, \mathrm{Msign}, \mathrm{Mverify})$ and $(\mathsf{Ukeygen}, \mathsf{Usign}, \mathsf{Uverify})$, under the hiding property of the commitment scheme, and the extractability and witness-indistinguishability of the proofs of knowledge, there exists a p.p.t. Sim such that for all p.p.t. Env:

$$\mathsf{Env}\|\mathsf{PSM}(\mathrm{M}, \tilde{\mathrm{P}}, \mathrm{U}) \equiv \mathsf{Env}\|\mathsf{Sim}\|\mathsf{Ideal}(F_{\mathsf{PSM}}, D_{\mathrm{M}}, \tilde{D}_{\mathrm{P}}, D_{\mathrm{U}}).$$

Claim (Security of PSM Against Corrupted User). Under the unforgeability of the signature schemes $(\mathrm{Mkeygen}, \mathrm{Msign}, \mathrm{Mverify})$ and $(\mathsf{Pkeygen}, \mathsf{Psign}, \mathsf{Pverify})$, under the binding property of the commitment scheme, and under the extractability and zero-knowledge property of the proofs of knowledge, there exists a p.p.t. Sim such that for all p.p.t. Env:

$$\mathsf{Env}\|\mathsf{PSM}(\mathrm{M}, \mathrm{P}, \tilde{\mathrm{U}}) \equiv \mathsf{Env}\|\mathsf{Sim}\|\mathsf{Ideal}(F_{\mathsf{PSM}}, D_{\mathrm{M}}, D_{\mathrm{P}}, \tilde{D}_{\mathrm{U}}).$$

We prove similar claims for our deposit and rebate extensions.

Claim (Security of PBR Against Corrupted Provider). Under the unforgeability of the signature schemes $(\mathsf{Mkeygen}, \mathsf{Msign}, \mathsf{Mverify})$ and $(\mathsf{Ukeygen}, \mathsf{Usign}, \mathsf{Uverify})$, under the hiding property of the commitment scheme, and the zero-knowledge property of the proofs of knowledge,[10] there exists a p.p.t. $\mathsf{Sim}_{\tilde{\mathrm{P}}}$ such that for all p.p.t. Env:

$$\mathsf{Env}\|\mathsf{PBR}(\mathrm{M}, \tilde{\mathrm{P}}, \mathrm{U}) \equiv \mathsf{Env}\|\mathsf{Sim}_{\tilde{\mathrm{P}}}\|\mathsf{Ideal}(F_{\mathsf{PSM}}, D_{\mathrm{M}}, \tilde{D}_{\mathrm{P}}, D_{\mathrm{U}}).$$

Proof. We describe a series of game transformations that gradually change the cryptographic payload of messages as they are sent by the real protocol into the cryptographic payload of messages of the simulation. In each step the new game is shown to be computationally indistinguishable from the previous one.

– **Game 0 - Game 3:** These game transformations are unchanged from Appendix A of [2]. **Game 3** corresponds to the execution of the real-world protocol, except that public keys pk_{M} and pk_{U} are replaced by other keys pk'_{M} and pk'_{U} from the same distribution, and that the game aborts if Env sends a message-signature pair (m, s) verifiable under public key pk'_{M} or pk'_{U} but for which Env did not receive a signature on message m verifiable under public key pk'_{M} or pk'_{U} respectively.

  Lemma 5. *Under the unforgeability of the signature schemes* $(\mathsf{Mkeygen}, \mathsf{Msign}, \mathsf{Mverify})$, $(\mathsf{Ukeygen}, \mathsf{Usign}, \mathsf{Uverify})$ *the difference* $|\Pr[\mathbf{Game\ 3}]-\Pr[\mathbf{Game\ 0}]| = \nu_{1-2}(\kappa)$.

– **Game 4:** This game proceeds as **Game 3**, except that the user keeps two different balances: $balance$ and $balance_c$: $balance$ is used for all checks, while $balance_c$ is the value in commitment $c_{balance}$. Commitments c_{price_i} are replaced by commitments to 0, and commitments c_{cons_i}, c_{other_i} are replaced by commitments to tuples $(0, other'_i)$ that map to 0 following Υ. The commitment c_{noise} in the payment message Q is replaced by a commitment to $noise' = fee$. The proofs π_i and π_{noise} are recomputed according to the Pay algorithm. The value of $balance$ is updated with respect to $noise$, while $balance_c$ and aux are updated with respect to $noise'$.

  Lemma 6. *Under the assumption that the commitment scheme is perfectly hiding and that the non-interactive proofs of knowledge are witness indistinguishable the difference* $|\Pr[\mathbf{Game\ 4}] - \Pr[\mathbf{Game\ 3}]| = \nu_3(\kappa)$.

– **Game 5:** This game proceeds as **Game 4**, except that the proof π_{inc} is replaced by a simulated proof of the same statement.

  Lemma 7. *Under the zero-knowledge property of the proofs of knowledge the difference* $|\Pr[\mathbf{Game\ 5}] - \Pr[\mathbf{Game\ 4}]| = \nu_4(\kappa)$.

– **Game 6:** This game proceeds as **Game 5**, except that the commitment c_{inc} in the payment message D is replaced by a commitment to the largest inc' such that $balance + inc' \geq 0$ and $\exists wit : ((inc', wit), instance) \in R$. The value of $balance$ is updated with respect to inc, while $balance_c$ and aux are updated with respect to inc'.

[10] Note that the zero-knowledge property implies witness indistinguishability.

Lemma 8. *Under the assumption that the commitment scheme is perfectly hiding and that the non-interactive proofs of knowledge are witness indistinguishable the difference* $|\Pr[\mathbf{Game\ 6}] - \Pr[\mathbf{Game\ 5}]| = \nu_5(\kappa)$.

$\mathsf{Sim}_{\tilde{\mathrm{P}}}$ does the same aborts, commitments, and proofs as **Game 6**, and forwards and receives messages from $\mathcal{F}_{\mathsf{PBR}}$ as described in Listing 4. The strategy behind the simulation of **setup** and **initialization** is unchanged from Appendix A of [2]. The distribution produced in **Game 6** is identical to that of our simulation. By summation we have that $|\Pr[\mathbf{Game\ 6}]| \leq \nu_6(\kappa)$. □

Listing 4 Simulator $\mathsf{Sim}_{\tilde{\mathrm{P}}}$

$\mathsf{Sim}_{\tilde{\mathrm{P}}}$ is parameterized by R and Y and interacts with $\mathcal{F}_{\mathsf{PBR}}$ through the dummy party $\tilde{D}_{\mathrm{P}}$ and with Env through the corrupted party $\tilde{\mathrm{P}}$. $\mathcal{F}_{\mathsf{REG}}$ is simulated as for the PSM scheme.

On $(\mathtt{Policy}, \Upsilon)$ from Env through $\tilde{\mathrm{P}}$ **(Initialization)**
- extract Υ; if $\Upsilon \notin Y$, abort
- if $\mathsf{VerifyPolicy}(pk_{\mathrm{P}}, \Upsilon_s) = 1$, store Υ_s, and send $(\mathtt{Policy}, \Upsilon)$ to $\mathcal{F}_{\mathsf{PBR}}$ through $\tilde{D}_{\mathrm{P}}$

On $(\mathtt{Deposit}, instance)$ from $\mathcal{F}_{\mathsf{PBR}}$ through $\tilde{D}_{\mathrm{P}}$ **(Deposit)**
- pick the largest inc such that $\exists wit : ((inc, wit), instance) \in R$
- compute a commitment $(c_{inc}, open_{inc}) = \mathsf{Commit}(par_c, inc)$ and a simulated non-interactive proof π_{inc}
- let $D = (\pi_{inc}, c_{inc})$ and $aux' = (balance + inc, open_{balance} + open_{inc}, c_{balance} \otimes c_{inc})$
- set $balance$ += inc and $aux = aux'$ and send $(D, instance)$ to Env through $\tilde{\mathrm{P}}$

On $(\mathtt{Pay}, from, until, fee)$ from F_{PBR} through $\tilde{D}_{\mathrm{P}}$ **(Payment)**
- set $noise = fee$ and $cons'_i = 0$ and pick $other'_i$ such that $\Upsilon(cons'_i, other'_i) = 0$
- for $i = from$ to $until$, run $\mathrm{SC}_i \leftarrow \mathsf{SignConsumption}(sk_{\mathrm{M}}, par_c, cons'_i, other'_i, i)$
- create a table T_{Sim} with the signed consumptions SC_i
- run $(\mathrm{Q}, aux') \leftarrow \mathsf{Pay}(sk_{\mathrm{U}}, par_c, \Upsilon_s, T_{\mathsf{Sim}}, noise, aux)$
- let $aux = aux'$; send $(\mathrm{Q}, from, until)$ to Env through $\tilde{\mathrm{P}}$.

Claim (Security of PBR Against Corrupted User). Under the unforgeability of the signature schemes $(\mathrm{Mkeygen}, \mathrm{Msign}, \mathrm{Mverify})$ and $(\mathrm{Pkeygen}, \mathrm{Psign}, \mathrm{Pverify})$, under the binding property of the commitment scheme, and under the extractability property of the proofs of knowledge, there exists a p.p.t. $\mathsf{Sim}_{\tilde{\mathrm{U}}}$ such that for all p.p.t. Env:

$$\mathsf{Env}\|\mathsf{PBR}(\mathrm{M}, \mathrm{P}, \tilde{\mathrm{U}}) \equiv \mathsf{Env}\|\mathsf{Sim}_{\tilde{\mathrm{U}}}\|\mathsf{Ideal}(F_{\mathsf{PSM}}, D_{\mathrm{M}}, D_{\mathrm{P}}, \tilde{D}_{\mathrm{U}}).$$

Proof. We describe a series of game transformations that gradually change the cryptographic payload of messages as they are sent by the real protocol into the cryptographic payload of messages of the simulation. In each step the new game is shown to be computationally indistinguishable from the previous one.

– **Game 0 - Game 4:** These game transformations are unchanged from Appendix B of [2]. **Game 4** corresponds to the execution of the real-world protocol, except for the following changes: The public keys pk_{M} and pk_{P} and the commitment parameters par_c are replaced by other keys pk'_{M} and pk'_{P} and $par_c{}'$ from the same distribution. **Game 4** aborts if the witnesses of the proofs π_i included in the payment messages Q cannot be extracted, or if these witnesses contain a message signature pair $(\langle cons, other, price\rangle, \mathrm{sp})$ that was not sent to Env. **Game 4** also aborts if any of the message-signature pairs $(\langle i, c_{cons}, c_{other}\rangle, \mathrm{sc})$ in the payment message Q was not sent to Env.

Lemma 9. *Under the proof of knowledge property of the proof system and the unforgeability of the signature schemes* $(\mathsf{Mkeygen}, \mathsf{Msign}, \mathsf{Mverify})$*, and* $(\mathsf{Pkeygen}, \mathsf{Psign}, \mathsf{Pverify})$ *the difference* $|\Pr[\mathbf{Game\ 4}] - \Pr[\mathbf{Game\ 0}]| = \nu_{1-3}(\kappa)$.

– **Game 5**: This game proceeds as **Game 4**, except that it aborts if the witnesses of the proofs π_{inc} and π_{noise} cannot be extracted.

  Lemma 10. *Under the proof of knowledge property of the proof system the difference* $|\Pr[\mathbf{Game\ 5}] - \Pr[\mathbf{Game\ 4}]| = \nu_4(\kappa)$.

– **Game 6**: This game proceeds as **Game 5**, except that it aborts if Env sends a payment message Q in which $(fee, open_{fee})$ is a correct opening of commitment $c_{noise} \otimes \left(\otimes_{i=1}^{N} c_{price_i}\right)$, but $fee \neq noise + \sum_{i=from}^{until} price_i$ where $noise$ and $price_i$ are taken from the witnesses of π_{noise} and π_i in Q.

  Lemma 11. *Under the binding property of the commitment scheme, the difference* $|\Pr[\mathbf{Game\ 6}] - \Pr[\mathbf{Game\ 5}]| = \nu_5(\kappa)$.

$\mathsf{Sim}_{\tilde{\mathrm{U}}}$ performs all the changes in **Game 6**, and forwards and receives messages from $\mathcal{F}_{\mathsf{PBR}}$ as described in Listing 5. The strategy behind the simulation of **setup**, **initialization**, and **consumption** is unchanged from Appendix B of [2]. The distribution produced in **Game 6** is identical to that of our simulation. By summation we have that $|\Pr[\mathbf{Game\ 6}]| \leq \nu_6(\kappa)$. □

Listing 5 Simulator $\mathsf{Sim}_{\tilde{\mathrm{U}}}$

$\mathsf{Sim}_{\tilde{\mathrm{U}}}$ is parameterized by R and Y and interacts with $\mathcal{F}_{\mathsf{PBR}}$ through the dummy party $\tilde{D}_{\mathrm{U}}$ and with Env through corrupted party $\tilde{\mathrm{U}}$. $\mathcal{F}_{\mathsf{REG}}$ is simulated as for the PSM scheme.

On $(\mathtt{Policy}, \Upsilon)$ from $\mathcal{F}_{\mathsf{PBR}}$ through $\tilde{D}_{\mathrm{U}}$ **(Initialization)**

- run $\Upsilon_s \leftarrow \mathsf{SignPolicy}(sk_{\mathrm{P}}, \Upsilon)$ and send Υ_s to Env through $\tilde{\mathrm{U}}$.

On $(\mathtt{Consume}, cons, other)$ from $\mathcal{F}_{\mathsf{PBR}}$ through $\tilde{D}_{\mathrm{U}}$ **(Consumption)**

- increment d_{M}, run $\mathrm{SC} \leftarrow \mathsf{SignConsumption}(sk_{\mathrm{M}}, par_c, cons, other, d_{\mathrm{M}})$
- send (SC) to Env through $\tilde{\mathrm{U}}$.

On $(D, instance)$ from Env through $\tilde{\mathrm{U}}$ **(Deposit)**

- run $(c'_{balance}, b) \leftarrow \mathsf{VerifyDeposit}(par_c, D, c_{balance}, instance, R)$.
- if $b = 1$ and none of the conditions described in **Game 5** are fulfilled, set $c_{balance} = c'_{balance}$, extract inc, wit from π_{inc} and send $(\mathtt{Deposit}, (inc, wit), instance)$ to $\mathcal{F}_{\mathsf{PBR}}$ through $\tilde{D}_{\mathrm{U}}$.

On $(\mathrm{Q}, from, until)$ from Env through $\tilde{\mathrm{U}}$, **(Payment)**

- run $(fee, c'_{balance}, b) \leftarrow \mathsf{VerifyPayment}(pk_{\mathrm{M}}, pk_{\mathrm{U}}, pk_{\mathrm{P}}, par_c, \mathrm{Q}, c_{balance}, from, until)$.
- if $b = 1$ and none of the conditions described in **Game 4**, **Game 5**, and **Game 6** are fulfilled, set $c_{balance} = c'_{balance}$, extract $noise$ from π_{noise} and send $(\mathtt{Payment}, from, until, noise)$ to $\mathcal{F}_{\mathsf{PBR}}$ through $\tilde{D}_{\mathrm{U}}$.

6 Conclusions

Our PBR protocol allows the user to add random noise to the final bill, to hide usage patterns that could otherwise be deduced from the fee. The rebate protocol supports deposits, anonymous payments using e-cash, and negative bill noise, while ensuring that the funds paid always cover the cost of consumption. The use of noise, however, comes at a cost, as it is money that the user has to pay upfront as a deposit and cannot invest elsewhere.

Consequently, we adapt the differential privacy framework to study how much noise is needed to protect specific consumption windows at different security levels. The differential privacy framework protects users against worse case outcomes – we leave as an open problem crafting more economical noise regimes to protect privacy by making further assumption about the users' typical behavior.

Acknowledgment

We would like to thank Claudia Diaz, Carmela Troncoso, Cedric Fournet, and Jorn Lapon for discussions and comments that were most helpful in the preparation and presentation of this work.

References

1. Balasch, J., Rial, A., Troncoso, C., Preneel, B., Verbauwhede, I., Geuens, C.: Pretp: Privacy-preserving electronic toll pricing. In: USENIX Security Symposium, USENIX Association (2010) 63–78
2. Rial, A., Danezis, G.: Privacy-preserving smart metering. Technical Report MSR-TR-2010-150, Microsoft Research (November 2010)
3. Lipner, S.B.: A comment on the confinement problem. In: Proceedings of the fifth ACM symposium on Operating systems principles. SOSP '75, New York, NY, USA, ACM (1975) 192–196
4. Dwork, C.: Differential privacy. In Bugliesi, M., Preneel, B., Sassone, V., Wegener, I., eds.: ICALP (2). Volume 4052 of Lecture Notes in Computer Science., Springer (2006) 1–12
5. Dwork, C.: Differential privacy in new settings. In Charikar, M., ed.: SODA, SIAM (2010) 174–183
6. Ghosh, A., Roughgarden, T., Sundararajan, M.: Universally utility-maximizing privacy mechanisms. In: Proceedings of the 41st annual ACM symposium on Theory of computing. STOC '09, New York, NY, USA, ACM (2009) 351–360
7. Barak, B., Chaudhuri, K., Dwork, C., Kale, S., McSherry, F., Talwar, K.: Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In Libkin, L., ed.: PODS, ACM (2007) 273–282
8. Goldwasser, S., Micali, S., Rivest, R.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2) (1988) 281–308
9. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In Brickell, E.F., ed.: CRYPTO '92. Volume 740., Springer-Verlag (1992) 390–420
10. Schnorr, C.: Efficient signature generation for smart cards. Journal of Cryptology **4**(3) (1991) 239–252

11. Chaum, D., Pedersen, T.: Wallet databases with observers. In: CRYPTO '92. Volume 740 of LNCS. (1993) 89–105
12. Okamoto, T.: An efficient divisible electronic cash scheme. In Coppersmith, D., ed.: CRYPTO. Volume 963 of Lecture Notes in Computer Science., Springer (1995) 438–451
13. Boudot, F.: Efficient proofs that a committed number lies in an interval. In Preneel, B., ed.: EUROCRYPT. Volume 1807 of Lecture Notes in Computer Science., Springer (2000) 431–444
14. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In Desmedt, Y., ed.: CRYPTO. Volume 839 of Lecture Notes in Computer Science., Springer (1994) 174–187
15. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In Odlyzko, A., ed.: CRYPTO. Volume 263 of LNCS., Springer (1986) 186–194
16. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical Report TR 260, Institute for Theoretical Computer Science, ETH Zürich (March 1997)
17. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS. (2001) 136–145
18. Backes, M., Pfitzmann, B., Waidner, M.: The reactive simulatability (rsim) framework for asynchronous systems. Inf. Comput. **205**(12) (2007) 1685–1720
19. Kusters, R.: Simulation-based security with inexhaustible interactive turing machines. In: Computer Security Foundations Workshop, 2006. 19th IEEE, IEEE (2006) 12–320
20. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: In SCN 2002, volume 2576 of LNCS, Springer (2002) 268–289

A Generalized Differential Privacy

We observe that differential privacy can be defined not only for databases that differ in exactly one record, but for any two database accepted by K that are in some relation $R \subset \mathcal{D} \times \mathcal{D}$.

Definition 8. *A randomized function K gives ϵ-differential R-privacy if for all data sets $D_1, D_2 \in R$, and all $S \in \Sigma_{Image(K)}$,*

$$Pr[K(D) \in S | D = D_1] \leq \exp(\epsilon) \times Pr[K(D) \in S | D = D_2] \ .$$

The probability is taken over the randomness of K.

Existing differential privacy mechanisms can be adapted to meet this new notion by adjusting the sensitivity function to $\Delta_{f,R}$.

Definition 9. *The sensitivity of a function $f : \mathcal{D} \to R^n$ is the maximum distance between output values for pairs $(D_1, D_2) \in R$:*

$$\Delta_{f,R} = \max_{D_1, D_2 \in R} \|f(D_1) - f(D_2)\|_1$$

For $n = 1$ the sensitivity of f is the maximum difference $|f(D_1) - f(D_2)|$ between databases D_1, D_2 that are in relation R.

The fact that differential privacy can be generalized in this way is by itself not very surprising. The new notion, however, becomes interesting when one stops using differential privacy to define that an individual is not worse off no matter whether he participates in a database or not, but to define that he (or in fact everyone contributing to the database) is no worse off no matter whether he behaves in this particular way (D_1) or that particular way (D_2) – for all behaviors resulting in databases related by $(D_1, D_2) \in R$.

In particular our u-privacy notion can also be defined in the following way:

$$R_u = \{(D_1, D_2) | Dist_u(D_1, D_2) \leq 1\} \ .$$

Given this definition of differential privacy, one can also start with a privacy budget that fixes Δ_f to examine the amount of protection one can buy by looking at

$$R_{\Delta_f} = \{(D_1, D_2\} \mid \|f(D_1) - f(D_2)\|_1 \leq \Delta_f\} \ .$$