A privacy-preserving statistics marketplace using local differential privacy and blockchain: an application to smart-grid measurements sharing

Nikos Fotiou, Iakovos Pittaras, Vasilios A. Siris, George C. Polyzos

Mobile Multimedia Laboratory
Department of Informatics, School of Information Sciences and Technology
Athens University of Economics and Business, Greece

Priit Anton

Guardtime
Tallinn, Estonia

Abstract

Service providers usually require detailed statistics in order to improve their services. On the other hand, privacy concerns are intensifying and sensitive data is protected by legislation, such as GDPR. In this paper we present the design, implementation, and evaluation of a marketplace that allows "data consumers" to buy information from "data providers," which can then be used for generating meaningful statistics. Additionally, our system enables "system operators" that can select which data providers are allowed to provide data, based on filtering criteria specified by the data consumer. We leverage local differential privacy to protect the data provider's privacy against data consumers, as well as against system operators, and we build a blockchain-based solution for ensuring fair exchange, and immutable data logs. Our design targets use cases that involve hundreds or even thousands of data providers. We prove the feasibility of our approach through a proof-of concept implementation of a measurement sharing application for smart-grid systems.

Keywords: Auditability, Ethereum, Fair exchange, Smart contracts

Email addresses: {fotiou,pittaras,vsiris,polyzos}@aueb.gr (Nikos Fotiou, Iakovos Pittaras, Vasilios A. Siris, George C. Polyzos), priit.anton@guardtime.com (Priit Anton)

1. Introduction

We are living in a globalized, cyber connected society where users have a plethora of choices. In this highly competitive environment, service providers try to offer as much personalized and consumer-tailored services as possible. In order to achieve their goal, they seek access to user profiling information. However, increased privacy concerns, as well as legislation, such as EU's General Data Protection Regulation (GDPR), have made the collection of such information a thorny challenge. Of course this comes as no surprise, since such activities not only jeopardize users' privacy, but also, as we have recently witnessed, the collected information can be used for manipulating users' choices [1]. Hence, the research question "how can sensitive data be securely shared?" still remains open.

Traditionally, the collection of sensitive information has been protected using anonymization techniques. However, large-scale privacy breaches from supposedly anonymized datasets, such as those involving AOL [2] and Netflix [3], have questioned the ability of those techniques to effectively protect user privacy [4]. A more promising approach is the use of *differential privacy* [5]. The goal of differential privacy is to allow the extraction of statistics about a population of users without revealing any information about specific individuals. This is achieved with the addition of some "noise" to the statistics extraction process. The added noise guarantees that the contribution of a single individual to the calculated statistics is not "significant." In other words, differential privacy guarantees that the output of the statistics calculation process is only slightly impacted by the contributions of a single individual, hence meaningful information about these contributions cannot be extracted.

Most related systems consider the so-called "centralized" differential privacy approach, where a trusted 3^{rd} party collects user data adds the appropriate amount of noise, and releases privacy-preserving statistics. Although this approach produces accurate results even with few samples, the introduction of a

trusted entity is a significant security and privacy risk. Furthermore, there can be cases where regulations or laws prohibit such 3^{rd} parties.

In this paper, we propose a privacy-preserving solution that allows a "data consumer" to extract meaningful statistics from sensitive data protected using "local differential privacy." Local differential privacy enables "data providers" to add noise to their (sensitive) data by themselves, and share them with untrusted 3^{rd} party data consumers without jeopardizing their privacy. Additionally, our solution considers an intermediate entity, referred to as the "system operator," that coordinates the whole process and applies filtering rules. With our approach, and as opposed to the state of the art, data providers are protected even against "curious" system operators. This is achieved by having the data providers send their (noisy) data directly to the data consumers.

Nevertheless, our approach leads inevitably to some tussles. For instance, a data consumer may be tempted to not pay the required fee. Similarly, data providers may indicate their interest to participate in a data collection process, receive the corresponding fee, but refuse to provide the actual data. Finally, a system operator may incorrectly filter out the responses of certain data providers. In order to resolve these tussles we rely on the blockchain technology. In particular, we leverage Ethereum smart contracts to provide a privacy-preserving immutable log of operations that can be used for dispute resolution, as well as to provide "fair exchange" of data and service fees. The contributions of our work presented in this paper are:

- We allow a "data consumer" to specify a minimum number of "data providers" it wishes to receive statistics from: if fewer data providers participate in an exchange then the data consumer does not have to pay for a service fee but at the same time it cannot generate any statistics.
- We enable "system operators" to filter out data consumers based on some specified criteria, without having access to their (noisy) data.
- We enable "fair exchange" between a data consumer and many (hundreds or even thousands) data providers, i.e., with our solution a data consumer

can access the (noisy) data of all providers only after paying the required service fee.

- Even though our solution involves many stakeholders, the cost for using the blockchain technology is minimum since the smart contract records only the minimum information necessary for auditing.

In order to put our solution in context, we consider a smart-grid measurements sharing use case.

With the smart meter penetration rate reaching 42,5% by 2020 and 83,87% by 2024 in the EU-28 [6], metering data can be a valuable source of information to various service providers. Although many *Distribution System Operators* (DSOs) provide data hubs for metering data, the vast majority of them prohibit 3^{rd} party service providers (e.g., telecom operators, service partners, technical integrators, and other 3^{rd} parties that may provide added-value services to smart meter owners) from accessing the stored data, mainly due to privacy concerns. Using our approach, smart meters can add noise to their measurements and directly share them with 3^{rd} parties, alleviating the burden of maintaining sensitive data from DSOs.

The rest of the paper is organized as follows. In Section 2 we provide background information, as well as related work in this area. In Section 3 we give an overview of the proposed system, and we detail its design in Section 4. We present the evaluation of our system in Section 5. Finally, we discuss some properties of the system in Section 6, and we present our conclusions in Section 7.

2. Background and Related work

2.1. Smart contracts and fair exchange

Smart contracts are decentralized applications that are deterministically executed in a blockchain (e.g., Ethereum). Smart contracts are widely used for implementing *escrow* services for exchanging digital goods in a fair way [7]. In particular, such an escrow smart contract allows "buyers" to deposit digital currency, (a portion of) which is transferred to a "seller" when a proof of digital

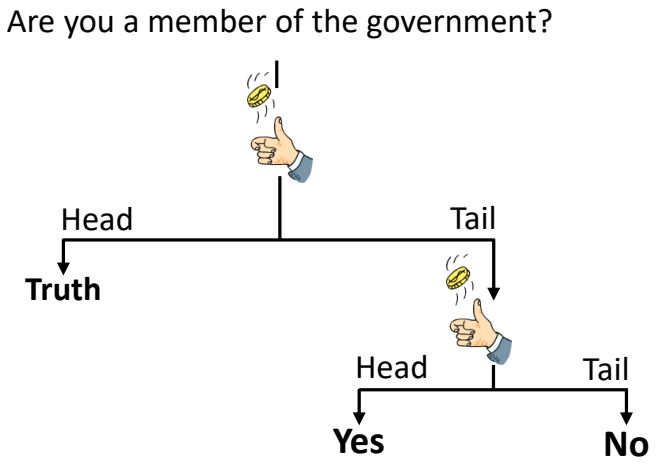


Figure 1: An example of randomized response. A respondent says the truth only if the flip of the coin comes up head, otherwise her response is determined by a second flip of the coin.

good exchange is presented to the contract. From a high-level perspective this exchange is implemented as follows: a buyer wishes to buy an information item i; the escrow smart contact is configured with the hash $h(i)$ of that item; the buyer deposits the appropriate amount of digital currency in the smart contact; then, the seller reveals to the smart contract an item i': the smart contact verifies that $h(i) = h(i')$, and if the verification is successful (which means $i' = i$) it transfers (a portion of) the deposited digital currency to the seller. Since the information recorded in a smart contract is public, the buyer can retrieve i'.

2.2. Local differential privacy

The main building block of local differential privacy mechanisms is an old survey technique known as "Randomized Responses" [8]. This technique involves a (sensitive) question that can be answered with a "Yes" or "No" (e.g., "Do you work for the government?"). A respondent flips a fair coin in secret, if the coin comes up heads she responds honestly, otherwise she flips the coin in secret again; this time if the coin comes up heads she responds with "Yes" and if it comes up tails she responds with "No". It can be seen that with this process, illustrated in Figure 1, 75% of the time the response will correspond

to the respondent's true answer. Moreover, and provided that she is not asked again the same question, the respondent's privacy is preserved since 50% of the time her response is generated at random. After a number of responses has been collected, the probability of a "Yes" response can be accurately estimated by calculating $minimum(0, 2 * (Y - 0.25))$, where Y is the proportion of "Yes" in the submitted responses [9].

RAPPOR [9], which has been developed by Google and is included with the Chrome browser, extends "Randomized Responses" surveys to allow for multiple possible answers. In its basic form ("Basic one-time RAPPOR," Section 2.1 of [9]), which is considered in this paper, RAPPOR involves a question that can be answered with a pre-defined number of choices (e.g., "Which of the following operating systems do you use: a) Linux b) Windows c) Mac OS d) Other "). A respondent constructs a bit vector of size equal to the number of choices (i.e., 4 in our example). The first position of the bit vector corresponds to the first choice, the second position to the second choice, and so forth. Then for each choice, she executes the randomized response algorithm (i.e, first she will respond to "do you use Linux?", then to "do you use Windows?" and so forth). If the output of the algorithm is "Yes", she fills the corresponding position of the vector with "1", otherwise with "0". After a number of bit vectors has been collected, the probability of the i^{th} choice is again estimated by calculating $minimum(0, 2 * (Y - 0.25))$, where Y is now the proportion of ones in the i^{th} position of the collected bit vectors (e.g., for the calculation of the probability of the choice "a) Linux" Y is set to the proportion of ones in the first position of all bit vectors).

2.3. Related work

Differential privacy has been considered by many research efforts including recommendation systems [10], data mining [11], crowd-sourcing [12], tools for performing network measurements [13], intelligent transportation systems [14], sensor network stream processing [15], and many others. These solutions have two key differences compared to our approach, firstly the differential privacy

mechanisms are applied by a trusted entity; this entity collects data from users, performs some computations, and extracts some privacy preserving statistics by adding "noise"; secondly, they assume that the entity that generates the statistics has also some sort of business relationship with the users, hence it is capable of "filtering" users based on pre-defined criteria (e.g., they can calculate the average of some metric only for users satisfying a location requirement). Unlike the above, our solution applies differential privacy at the data source, hence offers better privacy (however, this higher privacy comes at the cost of requiring more data inputs to achieve similar accuracy to that of centralized approaches). Secondly, our solution differentiates the entities that calculate a statistic from those that perform user filtering. This has the advantage that the entity responsible for filtering does not learn the calculated statistic. In order to illustrate the importance of this property consider the case of a university ranking authority; assume this authority wants to calculate how students rank their universities: whenever a user submits a response, the university can verify whether or not this user is a student, but it does not have access to the response submitted by the user nor to the calculated rank.

Recently, a number of research efforts have leveraged blockchain technology to implement "fair-exchange" of data, focusing at the same time on privacy preservation. Dimitriou and Mohammed [16] are using Bitcoin to perform fair exchange of smart-grid measurements. Nevertheless, with their solution, once the payment is made the data consumer has access to the actual user data, hence user privacy is not preserved. In contrast, with our solution user privacy is always protected. Another notable difference is that the solution in [16] performs an "one-to-one" fair exchange. Our approach supports a more complex model where many users encrypt their data: once a sufficient number of users have provided data, the data consumer makes the payment, and the (noisy) data of all users is revealed. Duan et al. [17] also use a blockchain-based approach where users stored their encrypted data in an Ethereum smart contract; then, a trusted entity decrypts the data, adds some noise (i.e., centralized differential privacy is used), and performs a fair exchange with a data consumer. Unlike this

approach, with our solution, users do not record their data in the blockchain (not even encrypted). This design choice is driven by the fact that our solution considers hundreds (or even thousands) of users: if users were recording their data in the smart contract the total transaction cost would be prohibitive.

A number of research efforts propose computations over sensitive information using cryptographic techniques, such as secure multi-party computation, or fully-homomorphic encryption (e.g., [18]). An advantage of these solutions, compared to our approach, is that they achieve accurate statistics. Of course, this is a double-edged sword: accurate statistics introduce privacy risks when the number of samples is small, or in cases where an attacker is able to monitor how the computed statistic "evolves" with the addition (or removal) of samples, which is the threat model of differential privacy. Moreover, these solutions have high complexity and impose significant computational overhead, hence they are not suitable for all use cases.

Traditional differential privacy mechanisms are based on adding noise sampled from a Laplace distribution to the aggregated data. In order for this noise to be added in a distributed manner, and hence avoid the need for a trusted aggregator, data providers should be able to communicate with each other. For example, Acs and Castelluccia [19] build a differential privacy solution for smart meter aggregate statistics (i.e., the same use case as in our paper). In order to avoid trusted aggregators, they organize smart meters in clusters and require the meters of each cluster to communicate with each other. Our solution does not require any communication among data providers, neither does it require that data providers know any information about the set of providers responding to a query (e.g., the total number of providers that send measurements). Similarly, a number of related works add noise locally, to a series of data, and the amount of noise is optimized based on the previous records (e.g. see [20]). Our solution assumes a different use case: data providers provide a single data point (e.g., consumption in a particular day). Instead of achieving local differential privacy using the Laplace distribution, our system uses the much simpler model of randomized responses. To this end, we build on RAPPOR, but other

randomized response systems can be used instead (e.g., a more recent system presented in [21]).

Gai et al. [22] use differential privacy to build a privacy preserving blockchain-based architecture for Industrial IoT. The goal of this architecture is to provide a privacy-preserving task allocation to "edge devices." The proposed architecture relies on a centralized entity referred to as the "Optimization Server" which is responsible for assigning tasks, collecting data, and adding "noise" to data using differential privacy techniques. In contrast, our approach uses local differential privacy, hence it does not need such a trusted centralized entity.

Fioreto et al. [23] apply differential privacy to obfuscate power line parameters without preventing however grid optimization algorithms. The proposed solution is very specific to the particular problem (power line parameters obfuscation) and it assumes a globally accessible model of the network to solve the problem. Our solution is simpler since it does not require any "global knowledge" from the data providers. Nevertheless, our solution cannot be used for the purposes of the problem discussed in [23]: the obfuscated response of a data consumer cannot be used as an input to an algorithm that requires high precision data (e.g., an optimization algorithm).

A number of works assume that smart meters cannot be trusted/modified and for this reason they use other devices operating in parallel to achieve local differential privacy. For example Zhao et al. [24] use batteries and through their charging and discharging functions they hide the energy usage patterns of the household. Our paper assumes that a smart meter is programmable and hence it can be modified to execute our local differential privacy algorithm.

Related work in this area investigates the integration of differential privacy into blockchain protocols. For example Hassan et al. [25] discuss how differential privacy can be integrated into blockchain building blocks. But this is completely orthogonal to our approach; we do not propose any modification to blockchain technology and we do not even store data in the blockchain (only hashes).

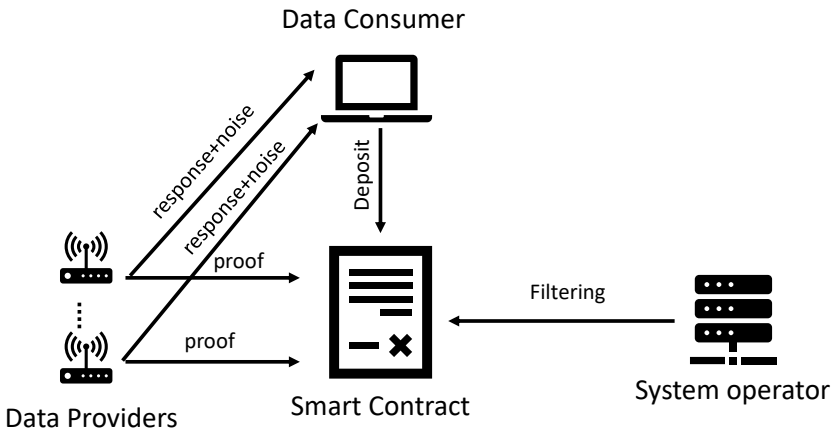


Figure 2: System overview

3. System overview and goals

Our system is composed of the following entities (Figure 2): A *data consumer* interested in extracting statistics about a population that meets certain criteria. Many *data providers* that can provide input used for generating the desired statistics. A *system operator* that is responsible for "filtering" the provided responses, i.e., for determining if the corresponding data provider meets the criteria specified by the data consumer. From a high-level perspective these entities interact as follows: A data consumer constructs a "query" which is answered by a data provider using local differential privacy. Such a query can be regarded as a vector $Q = \{c_1, c_2, ...c_n\}$, where $c_1, c_2, ...c_n$ are possible choices. Additionally, the data consumer specifies "criteria" that define which data providers are allowed to respond. Data providers "express their interest" to respond to a query. The system operator indicates which of the data providers that expressed interest meet the defined criteria. Then, those providers send their responses directly to the data consumer. Eventually, the data consumer extracts the desired statistics, compensates the system operator, which in turn compensates the appropriate data providers. The extracted statistics indicate the percentage of data providers that responded positively to each choice $Q[c_x] \forall x \in [1, n]$.

As we discuss in the following section, to achieve a high accuracy of the

extracted statistic, the number of responses should be of the order of hundreds or even thousands. Our system has the following properties:

- Nobody, not even the system operator can determine the exact response of a data provider.
- A data consumer cannot extract any statistics before paying for the provided services.
- A data consumer is required to pay only if a pre-agreed number of responses has been collected.

Our system achieves these goals through a blockchain-based solution. In particular, we design an Ethereum smart contract that can be used as an "immutable log" as well as a medium for "fair exchange."

Next we discuss the trust relationships that our system assumes. Firstly, a data consumer trusts data providers to answer honestly. This is the hardest trust assumption of our system, since due to the privacy-preserving properties of our solution nobody can tell which providers responded honestly. This level of trust can be (partially) achieved by implementing the provider-specific functionality in a Trusted-Platform Module (TPM). Further discussion of such a solution is out of the scope of this paper. Secondly, data consumers trust system operators to perform correct filtering. Similarly, it should be impossible for data consumers to verify that filtering was correctly applied, otherwise that would mean that they have access to sensitive information. Nevertheless, since the filtering rules and the filtering results are public, data providers can "fill a complaint" or "rate a system operator negatively" in case they are incorrectly excluded. Although we do not implement such mechanisms, in Section 5.3 we present dispute resolution tools that are enabled by the use of the Ethereum smart contract and can be used as building blocks for such systems. Thirdly, data consumers and data providers trust system operators to distribute the compensation paid by data consumers to data providers fairly. Again the tools

presented in section 5.3 can be used for building the appropriate reputation systems.

In the next section we describe in the detail the design of our solution. In order to better illustrate the concepts and their corresponding intuition we consider the use case of a smart-grid measurements sharing system. In this use case, the data providers are the deployed smart meters. A data consumer is a 3^{rd} party interested in extracting statistics about the grid, e.g., the average energy consumption in a particular city. The system operator is the smart meters' operator which holds information that is necessary for filtering the provided responses. We assume that the system operator cannot provide (or is not allowed to provide) the requested statistics.

4. Design and implementation

We assume that the system operator and the data providers are configured with a pre-shared secret key (psk). Furthermore, they are configured with a hash function $H(data)$, a keyed-hash message authentication code function $HMAC(key, data)$, and a symmetric encryption algorithm $E(key, data)$. Each data provider has an unlimited pool of Ethereum addresses and every time it responds to a query it uses a different one: the system operator always knows the current Ethereum address of each provider.

Data consumers should express the requested data in the form of a query with multiple choices. For example, suppose a data consumer is interested in learning the energy consumption per day of each smart meter, then a appropriate query could be "What is your energy consumption per date? a) 1kW b) 2kW c) 3kW ...". No matter the number of responses, there is always the chance that a user will respond randomly with probability 50%. Moreover, the number of responses per choice that will be estimated by the data consumer is not affected by the number of choices offered to data providers, hence the number of choices and their granularity are determined based on the needs of each use case. Finally, we record in the blockchain a cryptographic hash of a "configuration" file that

includes the query and the available choices, keeping this way the blockchain-related cost fixed and independent of the number of choices. Additionally, a data consumer can specify filtering rules for the specific query, e.g., "smart meters must be located in Athens, Greece." The protocols implemented by our architecture are discussed next.

4.1. Smart contract creation

With this protocol, a system operator and a data consumer agree on a "survey configuration" that includes (i) the query $Q = \{c_1, c_2, ...c_n\}$ that data providers should respond, (ii) the filtering rules, and (iii) the number of responses N_R that should be collected. The location of this configuration (e.g., a URL) as well as its hash are stored in a smart contract. Additionally, the system operator and the data consumer agree on two nonces $n1$ and $n2$. All data providers and the system operator (but not the data consumers) can derive an encryption key $sk = H(s1||s2)$, where $s1 = HMAC(psk, n1)$, $s2 = HMAC(psk, n2)$, and $||$ denotes concatenation. Finally, $s1$ is securely transmitted to the data consumer, and $n1$, $n2$, $H(s2)$, and a "service fee" are recorded in the smart contract.

4.2. Response commit

Data providers can retrieve the survey configuration from the specified location and verify its integrity. Any data provider wishing to respond to a query, prepares a "noisy" response R using local differential privacy. The response is constructed using the "basic one-time RAPPOR" algorithm (see section 2.2). In particular, R is a bit vector of size equal to the query Q. The value of each element $r_i \in R$ is equal to the output of the randomized response game (also discussed in Section 2.2) for the choice $c_i \in Q$, i.e., the value of the first element of R is the output of the randomized response game for the first choice in Q, and so forth. Then, the data provider derives the encryption key sk using the nonces $n1$ and $n2$ included in the smart contact and generates a ciphertext $C_R = E(sk, R)$, i.e., it encrypts the bit vector R with sk. Finally, it records

the hash of the generated ciphertext $H(C_R)$ in the smart contract. All hashes $H(C_R)$ are stored in the smart contract in a sorted map $M_{address \rightarrow H(C_R)}$ that maps the Ethereum address of the data provider to the corresponding $H(C_R)$.

4.3. Response filtering

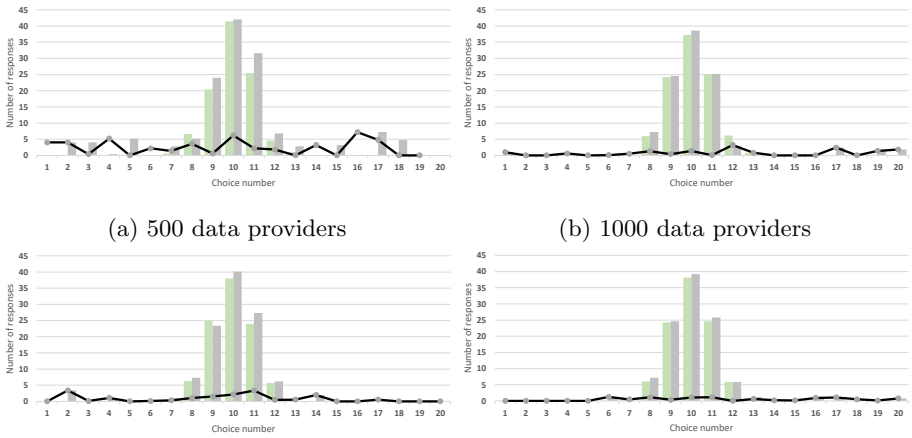
A system operator maintains a filter F. F is a dynamic bit vector whose size is increased by one every time an $H(C_R)$ is recorded in the map M of the smart contract. The value of a bit $f_i \in F$ is set to 1 if the i^{th} data provider that recorded a response in the smart contract meets the agreed filtering criteria otherwise it is set to 0. The implementation of the process for deciding about whether a response meets the filtering criteria is application specific. When the number of ones in F is equal to N_R (i.e., the number of the responses that can be accepted equals to the number of the required responses) the system operator generates the hash $H(F)$, records it in the smart contract, and makes F available to the data providers and to the data consumer (e.g., it publishes in a pre-agreed URL). At this point no new responses are accepted.

4.4. Response submission

All data providers that have committed a response (by submitting $H(C_R)$ to the smart contract) retrieve the filter F (from the system operator), as well as the map M (from the smart contract). Based on F a data provider can learn if it has been accepted to submit its response by executing the following process. It locates the position i in M that corresponds to its Ethereum address, and it examine the value of $f_i \in F$(i.e., the i^{th} element of F). If $f_i = 1$ then it is accepted and it sends i and the corresponding ciphertext C_R to the data consumer (it is reminded that $C_R = E(sk, R)$, i.e., C_R is the encryption of the noisy response using sk) .

4.5. Fair exchange

Upon receiving i and C_R from a provider, the data consumer verifies (a) the integrity of C_R using the hash $H(C_R)$ stored in the i^{th} position of the



(a) 500 data providers

(b) 1000 data providers

(c) 5000 data providers

(d) 10000 data providers

Figure 3: The impact of local differential privacy. Green axes show the real number of responses. Grey axes the estimated responses. The black line shows their difference: the closer the black line is to x axis the better.

map M (retrieved from the smart contract), and (b) that the data provider has been accepted to submit C_R by examining if the value of the i^{th} element of F (retrieved from the system operator) is 1. When it receives the agreed number of responses it deposits to the smart contract the appropriate amount of currency. Then the system operator records $s2$ to the smart contract; if the hash of the revealed $s2$ matches the hash $H(s2)$ stored in the smart contract, the contract transfers the deposit to the system operator. The system operator is then trusted to compensate the proper data providers accordingly. The data consumer can now reconstruct the data encryption key using $s1$ (received with the smart contract creation protocol), and $s2$, simply by calculating $H(s1||s2)$.

4.6. Results extraction

The data consumer constructs sk and uses it to decrypt the received C_R's. Each decrypted message is the noisy response of a data provider. Then, using the formula of Section 2.2 it calculates the probability of each choice $c_i \in Q$.

5. Evaluation

5.1. Local differential privacy efficiency

The accuracy of RAPPOR (and of local differential privacy in general) depends on the number of the submitted responses. In order to evaluate the accuracy of the extracted statistics, we perform the following experiment.[1] We create a number of data provider instances (the number of instances is a variable in our experiments) and we ask them to respond to a query that includes 20 possible choices. Each provider instance selects its response using a normal distribution with mean value 10 and deviation 2. Then each data provider submits its response. We then plotted (Figure 3) the real distribution of the responses (green bar), the extracted distribution based on the noisy responses (black bar), and their difference (black line): the closer to the horizontal axis the black line is, the more accurate the results are. The following diagrams concern experiments with 500, 1000, 5000, and 10000 data providers. It should be noted that the accuracy of the extracted results is not affected by the number of choices, neither by the distribution of the real responses.

5.2. Blockchain-based overhead

We implemented[2] a smart contract that provides the functionality described in Section 4 and we deployed and tested it in the Rinkeby Ethereum test network.[3] The following table shows the cost measured in units of "gas" for deploying our smart contract, as well as for invoking its functions.

The fiat cost included in this table is based on the prices retrieved from https://ethgasstation.info on 1 Jun. 2021. The cost of each operation is not affected by the number of data providers neither by the number of available choices.

[1] Our implementation, as well as instructions for replicated the experiments can be found at https://github.com/mmlab-aueb/rappor

[2] The implemented system can be found in https://github.com/mmlab-aueb/Privacy-and-Data-Sovereignty

[3] https://www.rinkeby.io/

Table 1: Smart contract costs

Operation	**Cost measured in gas**	**Cost measured in fiat**
Contract Deployment	660809	$32,5
Record $H(C_R)$	74537	$3,66
Record $H(F)$	63309	$3,11
Make deposit	23642	$1,16
Reveal $s2$ and transfer deposit	36269	$1.78

5.3. Dispute resolution

A key role of the smart contract in our system is to act as an immutable, append-only log where the outcomes of a number of hash functions are recorded. These records can be latter used with a dispute resolution mechanism.

Each data provider records in the smart contract $H(C_R)$. Then data providers send C_R to data consumers. In case C_R is unreadable, e.g., it cannot be decrypted with sk, the data consumer can use $H(C_R)$ in a dispute resolution process: since $H(C_R)$ is recorded in the blockchain, a data provider cannot deny it sent C_R.

Similarly, system operators record $H(F)$. A data consumer not included in the filter F can file a complain. Since $H(F)$ is recorded in the blockchain a system operator cannot deny that it used F. $H(F)$ can also be used by dispute resolution system that handles cases where a data provider is included in F but it did not receive the appropriate compensation.

5.4. Privacy properties of RAPPOR

We now discuss the privacy properties of RAPPOR. We consider the case of a question that includes n choices, i.e., $Q = \{c_1, c_2, ...c_n\}$ and the real responses follow a uniform distribution, hence for every provider $P_{(x=C)} = 1/n$, where C is the choice of the provider. The probability that an attacker with no additional information guesses the choice C of a provider is $P_{guess} = 1/n$. We now consider an attacker that can access a randomized response R of a provider, generated

using RAPPOR. The attacker tries to guess C by selecting at random an element of $r_i \in R$ whose value is 1. The probability that $r_i = 1$ is given by calculating the following formula.

$$P_{(r_i=1)} = \frac{1}{n} \times 0.5 + 0.25 \tag{1}$$

This happens because an element is 1 either if it corresponds to the actual choice C of the provider and the output of the randomized game is *true* (which has probability $\frac{1}{n} \times 0.5$) or the output of the randomized game is "say 1" (which occurs with probability 0.25). Therefore the numbers of $1s$ in R is $\approx P_{(r_i=1)} \times n$. Furthermore, the probability that an element of this set is C is 75%. We define the probability that this attacker guess correctly C as P_{RAPPOR} and we define the advantage of this attacker as $Adv = P_{RAPPOR}/P_{guess}$. Figure 4 shows these probabilities and the advantage of the attacker which approximates 3. This means that even though the P_{RAPPOR} decreases as the number of responses increases, it is ≈ 3 times bigger than P_{guess}. An intuitive explanation for this is that as the number of choices increases, $P_{(r_i=1)}$ approximates 0.25, hence, the number of $1s$ in R approximates $0.25 \times n$ therefore, $P_{RAPPOR} \approx \frac{1}{0.25 \times n} \times 0.75 = \frac{3}{n} = 3 \times P_{guess}$

An interesting trade-off that we can consider is to assume a "non-fair" coin for the first round of the randomized response game. For example a coin that decides that a user will say the truth 20% of the times (as opposed to 50%) decreases the advantage of the attacker to 1.5 (since, as the number of choices increases, the number of $1s$ in R approximates $(0.8 \times 0.5) \times n$ and the probability that an element of this set is C is 60%). Figure 5 shows the difference between real distributions of the responses and the distribution based on the noisy responses for 10000 data providers and 20 choices, selected using the distribution presented in section 5.1, using a fair coin (black line) and a coin that decides that the user will say the truth 20% of the times. As it can be seen, by using the second coin we reduce the advantage of the attacker, but at the cost of lower accuracy.

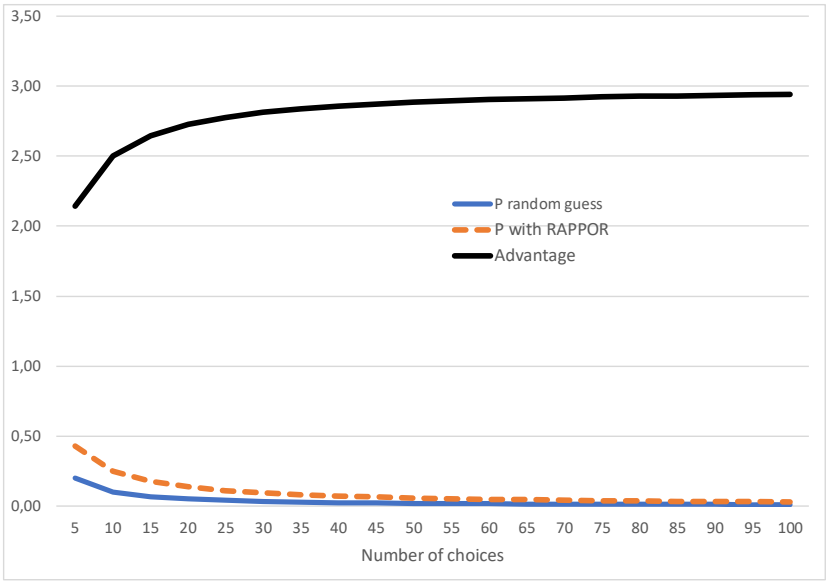


Figure 4: Probability of a successful random guess, and of a guess based on a RAPPOR response. The "Advantage" line shows the advantage of an attacker that has access to a RAPPOR response.

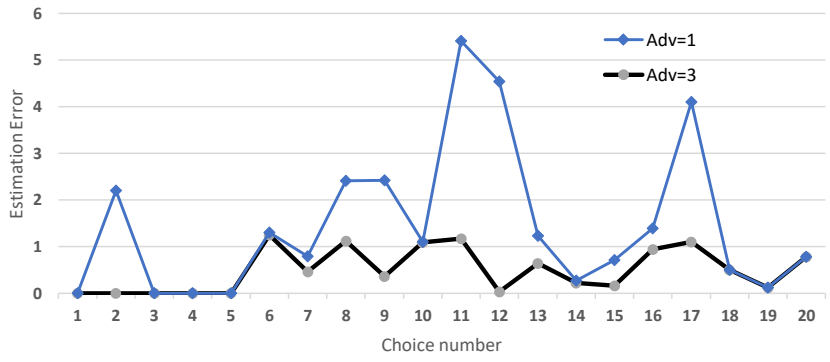


Figure 5: Error in estimation when different levels of attacker's advantage are considered.

5.5. Privacy-accuracy trade-off

The threat model considered in differential privacy mechanisms is that of an attacker that can monitor how the computed statistic "evolves" with the addition (or removal) of samples. Solutions that provide accurate statistics fail to provide protection against this type of attackers. In order to illustrate this attack, we perform the following experiment: we consider providers that select uniformly and at random a choice from the set $0, 1, .., 19$ and securely commit their response to a trusted service; every time a provider commits a response, the service outputs the average of all committed responses. After 1000 providers have committed their responses, an attacker starts monitoring the extracted average. Then, a new provider commits a response, the new average is calculated, and the attacker tries to guess based on the new average what was the choice of the provider. We consider two cases: (a) the responses are committed without noise addition (e.g., a mechanism based on homomorphic encryption is used), hence the extracted results are accurate, and (b) noise is added to the responses using RAPPOR. We perform the same experiment with all possible choices the 1001^{st} provider can make. Figure 6 shows the choice estimated by the attacker. The horizontal axis of the diagram included in this figure corresponds to the correct choice. As it can be seen, when no noise is added the attacker can successfully guess the added choice. However, when RAPPOR is used this is not possible.

6. Discussion

Our system has the following properties:

It is fast. Data providers have only to record a single hash in the smart contract and then send a small message (in order of hundreds of bytes).

It is lightweight. The most computationally intensive operation that a data provider has to perform is the calculation of a hash, the calculation of a digital signature, and a symmetric key encryption. Similarly, a data consumer has only to perform hash calculations and multiplications, and a symmetric key

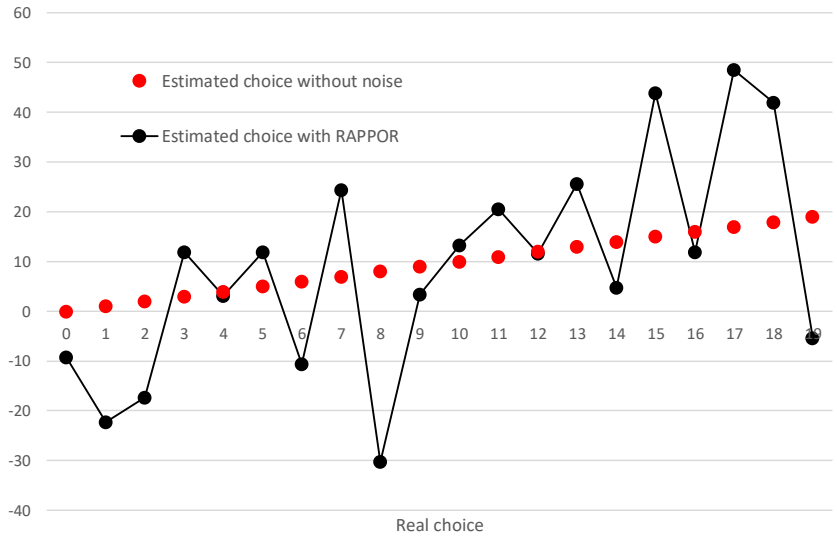


Figure 6: Guessed value when no noise is added to the response (red dots) and with RAPPOR (black dots). The horizontal axis corresponds to the correct choice.

decryption. All these computations can be performed fast even by constrained devices.

Many of the design choices in our system are driven by the need for supporting many data providers. For example, having the system operator distributing the service fee to the data providers, instead of the smart contract itself, may appear strange. However, implementing digital currency transfer to hundreds of users in a smart contract has prohibitive cost (and if the number of users is very big such an operation may not even be allowed). An alternative approach can be the implementation of a "withdrawal" function in the smart contract that eligible data providers can invoke, after the release of $s2$, and retrieve their compensation by themselves. Similarly, one may argue that the smart contract should filter the map M based on the filter provided by the system operator, nevertheless since M contains hundreds of records this is a very costly operation to implement in a smart contract.

A parameter of our system–as well as of any RAPPOR-based system–is the number of available choices per question. We discuss in Section 5.4 that, as the

number of choices increases, the advantage, Adv, of an attacker that has access to a RAPPOR response remains stable. Of course, the bigger the number of choices the better the privacy protection for providers since the probability of a successful guess is $Adv \times P_{guess}$, where P_{guess} is the probability of randomly guessing the choice of a provider; as the number of choices increases P_{guess} decreases. Moreover, as the number of choices increases the accuracy of the extracted results also increases. Nevertheless, and as we show in Section 5.1, the use of RAPPOR introduces some error in the extracted results which depends on the number of providers, e.g., it can be up to 5% with 500 providers and close to 1% with 10.000 providers. Therefore, the number of choices should be selected such that the distribution of the extracted results makes sense even with the error rate of RAPPOR. For example, in a scenario with 500 providers, offering many choices would result in a distribution where the most popular choices are selected by $\approx 5\%$ of the providers, will make the real results indistinguishable from the error rate.

Our system can be modified to support even higher privacy or greater decentralization. The former case is related to the fact that a data consumer can perform some form of data provider tracking based on the provider's network address (recall that the Ethereum address of each provider changes whenever they commit a response). Therefore, an alternative could be that the data providers send their encrypted responses to the system operator, and then the system operator to forward all of them to the data consumer. On the other side, it can be argued that some type of filtering can be performed by the data consumer (or even the smart contract) if data consumers can prove some of their properties (e.g., using *Verifiable Credentials* [26] that include their location). In that case the role of the system operator would be reduces and limited, e.g., to issuing the VC for verifying the data consumer's location or any other property used for filtering.

Local differential privacy systems–such as ours–are susceptible to "longitudinal" attackers, i.e., attackers that have access to multiple reports over time from the same user. RAPPOR protects users from this attack by introducing a

"memoization" step [9]. With this step, when a user is offered the same choice, the user responds with the same, permanently stored response. Nevertheless, defining whether two choices are the same may require human intervention, e.g., the choices "consumption at night is between 1kW and 5kW", and "consumption from 6pm to 4am is between 1kW and 5kW" can be considered the same. We postulate that since the service provider knows all the questions that have been asked it is its role to inform data consumers if a choice is the same to one previously offered.

Another aspect that affects the deployability of our system is the cost of transacting with the public blockchain. As it can be seen from Table 1 currently the most expensive operation (apart from deploying a smart contract) has cost \$3,66. These costs may limit the applicability of our approach for certain use cases. Similarly, these costs have high fluctuation, which is another problem in itself, since they depend on the price of ETH, i.e., the Ethereum digital currency.

7. Conclusions and future work

In this paper we presented a privacy-preserving marketplace for sensitive data. Our solution allows data consumers to "purchase" noisy data that can be used for extracting meaningful statistics. The privacy of data providers is protected against all entities of our system using local differential privacy. Our solution allows intermediate service providers that can filter out data providers without having access to the provided data neither to the extracted statistics. Our solution uses a blockchain-based approach for providing an immutable log, as well as for implementing fair exchange. Although our system may involve many stakeholders, the blockchain-based overhead is small and independent of the number of data providers.

The use of local differential privacy means that nobody can tell if a data provider responded to a query honestly. Therefore, increased privacy comes at the cost of exposure to malicious data providers. An interesting approach for mitigating this issue is the execution of the data provider specific functionality

in a Trusted Platform Module (TPM): the use of TPMs will be considered in our future work.

Ethereum smart contracts enable fair exchange, as well as immutable logs. Although our design keeps interactions with the blockchain to a minimum, and it has a fixed blockchain-based cost, no matter the number of the stakeholders, the overhead, the complexity, and the monetary cost introduced by this technology cannot be ignored. To this end, we are studying alternatives based on private, consortium-based blockchain systems.

Acknowledgment

This research was supported by the EU funded Horizon 2020 project SOFIE (Secure Open Federation for Internet Everywhere), under grant agreement No. 779984.

References

[1] D. Susser, B. Roessler, H. Nissenbaum, Online manipulation: Hidden influences in a digital world, Georgetown Law Technology Review, Forthcoming.

[2] Z. J. T. Barbaro M., A face is exposed for aol searcher no. 4417749 (2006). URL `https://www.nytimes.com/2006/08/09/technology/09aol.html`

[3] A. Narayanan, V. Shmatikov, Robust de-anonymization of large sparse datasets, in: 2008 IEEE Symposium on Security and Privacy (SP 2008), 2008, pp. 111–125.

[4] P. Ohm, Broken promises of privacy: Responding to the surprising failure of anonymization, UCLA l. Rev. 57 (2009) 1701.

[5] C. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating noise to sensitivity in private data analysis, in: S. Halevi, T. Rabin (Eds.), Theory of Cryptography, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 265–284.

[6] E. C. D. Energy, Benchmarking smart metering deployment in the eu 28, revised final report (2019).
URL https://www.vert.lt/SiteAssets/teises-aktai/EU28%20Smart%20Metering%20Benchmark%20Revised%20Final%20Report.pdf

[7] S. Dziembowski, L. Eckey, S. Faust, Fairswap: How to fairly exchange digital goods, CCS '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 967–984.

[8] S. L. Warner, Randomized response: A survey technique for eliminating evasive answer bias, Journal of the American Statistical Association 60 (309) (1965) 63–69.

[9] U. Erlingsson, V. Pihur, A. Korolova, Rappor: Randomized aggregatable privacy-preserving ordinal response, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, ACM, New York, NY, USA, 2014, pp. 1054–1067.

[10] F. McSherry, I. Mironov, Differentially private recommender systems: Building privacy into the netflix prize contenders, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 627–636.

[11] A. Friedman, A. Schuster, Data mining with differential privacy, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '10, Association for Computing Machinery, New York, NY, USA, p. 493–502.

[12] J. Hamm, A. C. Champion, G. Chen, M. Belkin, D. Xuan, Crowd-ml: A privacy-preserving learning framework for a crowd of smart devices, in: 2015 IEEE 35th International Conference on Distributed Computing Systems, 2015, pp. 11–20.

[13] A. Mani, M. Sherr, Historε: Differentially private and robust statistics collection for tor., in: NDSS, 2017.

[14] F. Kargl, A. Friedman, R. Boreli, Differential privacy in intelligent transportation systems, in: Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 107–112.

[15] A. Friedman, I. Sharfman, D. Keren, A. Schuster, Privacy-preserving distributed stream monitoring., in: NDSS, 2014, pp. 1–12.

[16] T. Dimitriou, A. Mohammed, Fair and privacy-respecting bitcoin payments for smart grid data, IEEE Internet of Things Journal 7 (10) (2020) 10401–10417. doi:10.1109/JIOT.2020.2990666.

[17] H. Duan, Y. Zheng, Y. Du, A. Zhou, C. Wang, M. H. Au, Aggregating crowd wisdom via blockchain: A private, correct, and robust realization, in: 2019 IEEE International Conference on Pervasive Computing and Communications (PerCom, 2019, pp. 1–10.

[18] M. Burkhart, M. Strasser, D. Many, X. Dimitropoulos, Sepia: Privacy-preserving aggregation of multi-domain network events and statistics, in: Proceedings of the 19th USENIX Conference on Security, USENIX Security'10, USENIX Association, USA, 2010, p. 15.

[19] G. Ács, C. Castelluccia, I have a dream! (differentially private smart metering), in: T. Filler, T. Pevný, S. Craver, A. Ker (Eds.), Information Hiding, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 118–132.

[20] M. Hale, P. Barooah, K. Parker, K. Yazdani, 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, UrbSys'19, Association for Computing Machinery, New York, NY, USA, 2019, p. 33–42.

[21] C. Liu, S. Chen, S. Zhou, J. Guan, Y. Ma, A general framework for privacy-preserving of data publication based on randomized response techniques, Information Systems 96 (2021) 101648.

[22] K. Gai, Y. Wu, L. Zhu, Z. Zhang, M. Qiu, Differential privacy based blockchain for industrial internet-of-things, IEEE Transactions on Industrial Informatics 16 (6) (2020) 4156–4165. doi:10.1109/TII.2019.2948094.

[23] F. Fioretto, T. W. K. Mak, P. Van Hentenryck, Differential privacy for power grid obfuscation, IEEE Transactions on Smart Grid 11 (2) (2020) 1356–1366.

[24] J. Zhao, T. Jung, Y. Wang, X. Li, Achieving differential privacy of data disclosure in the smart grid, in: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014, pp. 504–512.

[25] M. Ul Hassan, M. H. Rehmani, J. Chen, Differential privacy in blockchain technology: A futuristic approach, Journal of Parallel and Distributed Computing 145 (2020) 50–74.

[26] Manu Sporny et al., Verifiable credentials data model 1.0 (2019).
URL https://www.w3.org/TR/verifiable-claims-data-model/