

Received 7 November 2022, accepted 2 December 2022, date of publication 8 December 2022, date of current version 14 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3227633

## RESEARCH ARTICLE

# Incentivizing Participation in Crowd-Sensing Applications Through Fair and Private Bitcoin Rewards

TASSOS DIMITRIOU<sup>1</sup>, (Senior Member, IEEE), AND ANTONIS MICHALAS<sup>2,3</sup>

<sup>1</sup>Computer Engineering Department, College of Engineering and Petroleum, Kuwait University, Kuwait

<sup>2</sup>Department of Computing Sciences, Tampere University, 33101 Tampere, Finland

<sup>3</sup>Research Institutes of Sweden (RISE), 16440 Stockholm, Sweden

Corresponding author: Tassos Dimitriou (tassos.dimitriou@ieee.org)

This work was supported by Kuwait University Research Grant EO 01/20.

**ABSTRACT** In this work we develop a rewarding framework that can be used to enhance existing crowd-sensing applications. Although a core requirement of such systems is user engagement, people may be reluctant to participate because sensitive information about them may be leaked or inferred from submitted data. The use of monetary rewards can help incentivize participation, thereby increasing not only the amount but also the quality of sensed data. Our framework allows users to submit data and obtain Bitcoin payments in a privacy-preserving manner, preventing curious providers from linking the data or the payments back to the user. At the same time, it prevents malicious user behavior such as double-redeeming attempts, where a user tries to obtain rewards for multiple submissions of the same data. More importantly, it ensures the *fairness* of the exchange in a completely trustless manner; by relying on the Blockchain, the trust placed on third parties in traditional fair exchange protocols is eliminated. Finally, our system is highly efficient as most of the protocol steps do not utilize the Blockchain network. When they do, only the simplest of Blockchain transactions are used as opposed to prior works that are based on the use of more complex smart contracts.

**INDEX TERMS** Crowd-sensing, participatory sensing, security and privacy, data reporting, incentives, rewarding mechanisms, *zkSNARKs*, bitcoin, blockchain.

## I. INTRODUCTION

Advances in sensing and networking have given rise to a new sensing paradigm called *participatory* or *mobile crowd-sensing* in which participants use their sensor-enabled devices to gather data of unprecedented quality and quantity about their immediate surroundings, leading to innovative applications such as intelligent transportation, environmental monitoring and urban sensing, assistive healthcare, and so on [1].

In such crowd-sensing applications, the service provider administers the data sharing infrastructure and recruits the people (and their devices) to gather data for the advertised sensing tasks. The collected data are then analyzed and made

available to the users or the broader public. It is this lack of a fixed sensing infrastructure and the ubiquitousness of WiFi and mobile Internet connectivity that gives crowd-sensing its unique advantage over traditional sensing paradigms.

From the viewpoint of the service providers, the key factor to the success of crowdsensing is user participation. However, collecting information from user devices has important privacy implications since contributed data may be strongly related to user activities and daily routines [2]. For example, sensed data may include visited locations or even personal data such as photos and videos. This in turn may have a negative impact on participation as users may be reluctant to endanger their personal lives without immediate benefits [3]. To this effect, monetary incentives could help attract a larger number of participants, thereby increasing the amount and the

The associate editor coordinating the review of this manuscript and approving it for publication was Luca Bedogni<sup>1</sup>.

quality of sensed data. Hence incentivizing users by means of *rewards* can be the only way to motivate user engagement and improve the quality of collected information.

The problem of incentivizing user participation has been studied in the literature (see [4] for a survey), however new design challenges arise regarding accountability and fairness. Accountability is important for two main reasons: First, only authenticated users should benefit from using the rewarding system. Second, malicious users should be prevented from abusing the system, for example by trying to obtain multiple rewards for the same data. Accountability, however, seems to contradict the desire for privacy; if user identities are not protected, task submissions and other contextual information like location data and participation history may result in serious privacy breaches.

Another important issue that has not been addressed adequately by prior research in the context of crowd-sensing is *fairness*. A malicious provider, for example, may refuse to pay the user after getting the data, even if this might hurt its reputation. On the other hand, if the provider pays first, nothing stops a user from “taking the money and run”, without releasing any sensed data. Thus, in the absence of a centralized authority that monitors the exchange of money-for-data and is responsible for resolving any conflicts that may arise due to failures or fraud, both parties can cheat the other. We address this problem by relying on the Blockchain network, thus eliminating the need for a trustworthy entity required in fair exchange protocols [5], [6].

*Contributions:* In this work,<sup>1</sup> we propose a rewarding framework that can be used to trade crowd-sensing data for bitcoins. We use the blockchain to safeguard the fairness of the exchange and we place no trust on the service provider or other third parties. Additionally, we rely on simple blockchain transactions as opposed to prior works that utilize the blockchain network to exchange the sensed data in addition to the rewards. This greatly reduces cost and enhances the efficiency of our protocol.

Our work is complementary to all approaches that aim at enhancing privacy when users report data and expect a reward. This is the case of crowd-sensing applications; every time sensed data has to be reported, our protocol ensures that a payment can be made to an ephemeral bitcoin address.

We have studied the privacy and security properties of our scheme and have proved that the rewarding framework is indeed privacy-friendly.

In summary, we make the following contributions:

- We develop a rewarding framework that can be easily integrated in existing crowd-sensing applications.
- Our protocol exhibits strong fairness since data is delivered *if and only if* an appropriate payment is received.
- The use of the blockchain helps eliminate trust in traditional fair-exchange protocols. Participants have a single

view of transactions and the system is publicly verifiable, hence no trust is placed on any third party.

- Our system relies on simple blockchain transactions, hence increasing efficiency and reducing cost. Additionally, the core steps of the protocol take place *offchain*. For completeness, we have implemented our protocol both in the Bitcoin and Ethereum blockchains, demonstrating the practicality of our approach.
- We use succinct zero-knowledge proofs (*zkSNARKs*) to ensure not only anonymity but also the well-formedness and efficiency of the various operations.
- We formally argue about the security and privacy aspects of the proposal. Malicious providers cannot identify users nor link protocol operations. Additionally, malicious users cannot issue/forgo rewards, nor they can claim more than one reward for the same data.

*Organization:* In the next section, we review related work in the context of crowd-sensing. In Section III, we present the system and threat model, and define the properties we expect from a secure data rewarding system. Section IV highlights the cryptographic tools used in our proposal while details of our protocol are presented in Section V. The protocol’s security and implementation aspects are discussed in Sections VI and VII, respectively. Finally, Section VIII concludes the paper.

## II. RELATED WORK

A number of works in the literature use the blockchain to exchange data and rewards. Tanas et al. [8] developed a framework that focuses mostly on the anonymity of payments since user data are sent in the clear to the data collector who must first perform a validation test on the data before payment. This may compromise fairness as a malicious collector may refuse to reward users once data are sent. A similar solution is proposed by Chen and Xue [9] in which a blockchain is used to record the exchange of data among participants. Feng and Yan [10] developed a blockchain-based crowd-sensing system, named MCS-Chain, to achieve decentralized trust management. However, none of these works has a focus on privacy.

CrowdBC [11] is a smart contract based solution which can be used by a requester to solicit a number of workers to solve a particular task. However, the system is neither private nor anonymous since the collected data and the user identities can leak to the blockchain network. In particular, as miners in CrowdBC collect and evaluate user data, they can collude with the collector and learn confidential data as well as affect fairness.

Lu et al. [12] developed ZebraLancer, a system similar to CrowdBC that tries to overcome data leakage and identity breach. The smart contract handles both worker submissions and payments by the collector, preventing dishonest behavior by either party. However, in the case of a cheating collector, the smart contract will distribute the budget evenly among users thus violating fairness since some users may have contributed more than others. Furthermore, as all information

<sup>1</sup>This paper improves and extends our prior work [7] that has appeared as a short paper in the IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS 2020).

(both user data and payments) goes through the smart contract this essentially turns the system into a centralized one.

The work of Duan et al. [13] is also based on smart contracts bearing monetary rewards along with hardware-assisted transparent enclaves to ensure correctness of data aggregation and sanitization of data. The system consists of consumers who post the sensing tasks as a smart contract, workers who send the data to the contract, and providers who perform data sanitization. Unfortunately, the threat model is very limited as the authors assume that data consumers and the service provider do not collude with each other since otherwise data confidentiality and user privacy is lost. The work of Dai et al. [14] also relied on the combination of smart contracts and secure hardware to develop a data trading ecosystem. However, recent works have shown that secure hardware can be compromised [15].

A characteristic of all smart contract-based solutions is that all data go through the smart contract, resulting in rather inefficient systems. FairSwap [16] attempts to overcome this issue by producing small proofs of misbehavior in case of a cheating user. When this happens, the smart contract penalizes the user, hence it plays the role of a TTP running on top of the blockchain. However, smart contract transactions induce direct monetary costs to the collector, thus affecting their practicality in large scale solutions. Our work overcomes these issues by having most of the protocol steps taking place *offchain*. Only the actual payment transaction uses the blockchain network, thus making the system highly efficient. Additionally, we do not rely on the blockchain network or the miners to store and verify data submissions as in past works. Doing so paves the way to *collusion* attacks in which malicious miners can collaborate with the collector to steal data or compromise user privacy.

A recent work that avoids the use of smart contracts is [17]. While the authors use rather simple blockchain transactions, both the workers and the collector have to go through many rounds of submissions in the blockchain, seriously affecting the efficiency of the protocol. More importantly, the worker IDs are visible and the collector obtains the data before the actual rewarding phase, thus violating anonymity and fairness. Wang et al. [18] proposed another incentive mechanism to reward users. Miners first verify the quality of the sensed data conforming to assessment criteria published by the server, then reward the users accordingly. However, since the miners get the data first, a malicious miner may forward these to the server thus undermining the fairness of the process. Finally, Delgado-Segura et al. [19] describe a protocol for fair data trading. However, fairness is only probabilistically enforced and the actual protocol follows a cut and choose approach: in order to convince the buyer that the required data is correct, a portion of it has to be revealed first. This increases the communication overhead of the protocol. Additionally, there is no way for the buyer to be sure about the utility of the data beforehand. Both issues are taken care by our protocol. A comparison of the most relevant

**TABLE 1. Comparison with existing protocols.**

Properties	[11]	[12]	[13]	[17]	[18]	[19]	This work
Centralized Smart contract	Yes	Yes	Yes	No	No	No	No
Privacy/Anonymity	- <sup>3</sup>	✓	-	-	-	-	✓
Fairness	- <sub>1,2,4</sub>	- <sub>1,4</sub>	-	-	- <sub>1,2</sub>	- <sub>5</sub>	✓

✓: Provided    ◊: Partially provided    -: Not provided

1. Possible collusion between malicious miners and  $\mathcal{UP}$ .
2. Miners evaluate solution beforehand.
3. User identities are handled by smart contract.
4. Solution encrypted with  $\mathcal{UP}$ 's public key.
5. Probabilistically enforced fairness.

works with emphasis on privacy and fairness can be found in Table 1.

In this work, the blockchain will be used to ensure the fair exchange between submitted data and respective rewards, eliminating the possibility of cheating by any party. Previous works on fairness using blockchain focus on fair purchase operations of a product. For example, Bentov and Kumaresan [20] use Bitcoin to enforce proper behavior of participants by means of a penalty mechanism, while Heilman et al. [21] design an unlinkable payment hub that allows participants to make fast, anonymous, off-blockchain payments through an untrusted intermediary. Campanelli et al. [22] achieve strong fairness for the case where a seller wants to be paid after proving that a service has been rendered, as opposed to selling a secret in Zero Knowledge contingent payments [23]. Our work is similar to those in the sense that the “secret” to be sold is the user data. However, care has to be taken to ensure that only authenticated users can benefit from the existence of rewards.

For token-based mechanisms that do not use the blockchain to reward users, the reader is referred to [24] and [25] and the references therein.

### III. SYSTEM MODEL

Our base architecture consists mainly of two entities: *users* that participate in sensing tasks and a *service provider* that collects data and rewards users for the data they provide. Users install sensing applications in their mobile devices and gather sensor readings which might include location, accelerometer data, pictures, sound samples, environmental data like temperature and pollution levels, and so on. Once they have data to report, they contact the application server which collects the results and organizes them in appropriate form for display to the general public or locally to the user devices.

A snapshot of this architecture is shown in Figure 1. Users  $\mathcal{U}$  transmit their sensed data to the service provider  $\mathcal{P}$  who defines the scope of the data collection campaign. The registration authority ( $\mathcal{RA}$ ) is responsible for registering and authorizing user participation (Step 1). The task server ( $\mathcal{TS}$ ) defines the sensing tasks, advertises those to the users and defines the rewards for the contributions they make. Typically, these entities are part of the back-end infrastructure

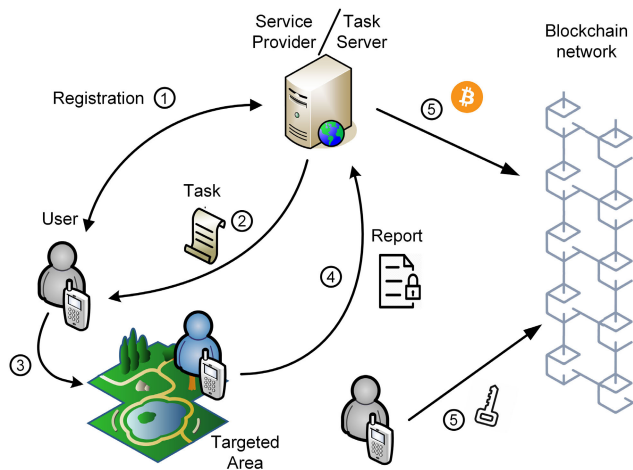


FIGURE 1. System architecture.

that supports the sensing process, starting with task definition, user registration and collection of sensing data, to final reporting and dissemination of results. Here, however, we will not distinguish between these entities as is customary in participatory sensing applications [2] since these entities may collude with each other to recover user information. Hence from a security point of view we will treat them as one, and without loss of generality, we will assume that the user is in possession of an authentic task (Step 2) and moves to the target area (Step 3) to start collecting data.

Our protocol operations emphasize on Steps 4 and 5, where users collect bitcoin payments for submitted data. To prevent a malicious provider from refusing to pay, the sensed data are sent encrypted. The decryption key will be released only when the user obtains a payment. Thus, Step 5 relies on the Blockchain to ensure that data is delivered *if and only if* an appropriate payment is made. This guarantees that no party can cheat, thus ensuring fairness.

In addition to the above, the provider needs to be convinced that (i) data have been evaluated correctly according to some public valuation function  $\mathcal{R}()$  which produces a value  $val = \mathcal{R}(m)$  equal to the utility of the data, and (ii) users cannot resubmit the same data again. This tension between accountability and anonymity is resolved in a methodical way, ensuring the privacy-respecting character of our protocol. Finally, we should note that with the exception of the key-for-money transaction, all protocol steps take place *offchain*. Table 2 summarizes the notation used throughout this work.

**A. MAIN OPERATIONS**

In the following we define the various operations expected from our system. For simplicity, only a high-level system interface is presented, not listing every possible input required to execute the protocols. These operations enable users to register, submit data, obtain awards, and so on.

- **Setup**( $1^\kappa$ ) is an algorithm executed by  $\mathcal{P}$  to setup the data reporting and payment system, where  $\kappa$  is the

TABLE 2. Key notation.

Notation	Definition
$\mathcal{U}, \mathcal{P}$	User and Provider
$taskID$	Identifier of sensing task
$\mathcal{R}()$	Public rewarding function
$m_1, \dots, m_n$	Sensed data in response to sensing task
$val = \mathcal{R}(m_1, \dots, m_n)$	Valuation of sensed data
$(pk_{\mathcal{U}}, sk_{\mathcal{U}}), (pk_{\mathcal{P}}, sk_{\mathcal{P}})$	Public-private key pairs of $\mathcal{U}$ and $\mathcal{P}$
$(pk_{\mathcal{U}}^e, sk_{\mathcal{U}}^e)$	Ephemeral key pair of $\mathcal{U}$
$K$	Key to encrypt sensed data by user
$C$	Encryption of sensed data using $K$
$H()$	Secure hash function
$\pi$	$zkSNARK$ proof
$\tau$	Double-redeeming tag

security parameter. The algorithm generates  $\mathcal{P}$ 's public and private key  $(pk_{\mathcal{P}}, sk_{\mathcal{P}})$  and the public parameters of the  $zkSNARK$  subsystem.

- **Register**( $\mathcal{U}, \mathcal{P}$ ) is executed by  $\mathcal{U}$  and  $\mathcal{P}$ .  $\mathcal{U}$  generates a long term public/private key pair  $(pk_{\mathcal{U}}, sk_{\mathcal{U}})$  and then  $\mathcal{P}$  registers  $pk_{\mathcal{U}}$  and other used identity information to prevent  $\mathcal{U}$  from mounting double-redeeming attacks. A second outcome of this protocol is an *ephemeral* key pair  $(pk_{\mathcal{U}}^e, sk_{\mathcal{U}}^e)$ . The private key  $sk_{\mathcal{U}}^e$  will be used by  $\mathcal{U}$  to submit sensing data and sign bitcoin transactions. The public key  $pk_{\mathcal{U}}^e$  (essentially a bitcoin address) will act as an *authorization* credential. Thus only authorized users can receive payments in a privacy-preserving manner. However, none of these keys can be linked to the long term public key of the user which is known to  $\mathcal{P}$ .
- **IdentDR**( $taskID, DB_{\mathcal{P}}$ ) is executed by  $\mathcal{P}$  to check if in its database  $DB_{\mathcal{P}}$  there exist two different submissions by the same user for a given task. This is to prevent *double-redemption* attempts in which users try to obtain more than one payments for the same sensing task.
- **Submit**( $\mathcal{U}, \mathcal{P}$ ) is executed between  $\mathcal{U}$  and  $\mathcal{P}$ .  $\mathcal{U}$  first picks a symmetric key  $K$  and encrypts the sensed data  $m$  to  $C = E_K(m)$ . Then, using an *anonymous* communication channel, it forwards  $C$  and the hash  $H(K)$  of the key to  $\mathcal{P}$ .  $\mathcal{U}$  also creates a proof  $\pi$  that (i) demonstrates knowledge of the key  $K$ , and (ii) that data in  $C$  is worth  $val$  bitcoins according to the rewarding function  $\mathcal{R}()$ . The data sent are also signed with the ephemeral signing key  $sk_{\mathcal{U}}^e$  to produce a signature  $\sigma_e$ .
- **Verify**( $\mathcal{P}, \sigma_e, C, h_K, \pi$ ). This protocol is executed by the provider to check the validity of the signature  $\sigma_e$  and the proof  $\pi$ . The signature attests to the fact that the user is authorized. If the proof  $\pi$  is also valid,  $\mathcal{P}$  posts a Bitcoin transaction  $T_{\mathcal{P} \rightarrow \mathcal{U}}$  to reward  $val$  bitcoins to the user who presents the key that hashes to  $h_K$ . This operation also uses **IdentDR** to check for double-redemption attempts.
- **Release**( $\mathcal{U}, K$ ). This is executed by  $\mathcal{U}$  once the user confirms that transaction  $T_{\mathcal{P} \rightarrow \mathcal{U}}$  has been posted on the blockchain network. The user posts a transaction  $T_{\mathcal{U} \rightarrow \mathcal{P}}$  that releases the key  $K$ .  $T_{\mathcal{U} \rightarrow \mathcal{P}}$  is signed with the private ephemeral key of the user to prevent *forwarding* attacks in the blockchain network.

*Definition 1 (Correctness):* A rewarding scheme is called correct if for all system parameters  $\mathcal{CRS}$  and key material created by  $\text{Setup}(1^k)$  and  $\text{Register}(\mathcal{U})$ , and for honest  $\mathcal{U}$  and  $\mathcal{P}$ , the following properties hold:

- *Correctness of issuing and verifications.* When issued in **Register** and used in **Submit** and **Verify**, ephemeral keys can be verified correctly.
- *Correctness of payment (fairness).* All data  $m$  worth value  $val$  under the rewarding function  $\mathcal{R}()$ , can be redeemed successfully under operations **Submit**, **Verify** and **Release**.

## B. SECURITY AND PRIVACY MODEL

Our threat model includes malicious providers and users. Malicious providers may try to collect sensed data without rewarding users. Alternatively, they may try to link user actions; if it is possible to link a reward to a payment or to an initial registration step then it might be possible to de-anonymize users. Malicious users on the other hand may try to get more from what they deserve for data submitted. To prevent user misbehavior we will demand that only authorized users can access the system. Our design will ensure that users remain anonymous, their actions unlinkable and yet they cannot misbehave. The necessary requirements are listed below:

- *Authorization.* The provider should be able to tell whether submitted data is coming from a legitimate, registered user, however without violating user privacy.
- *Confidentiality and integrity.* Reported data should be protected against eavesdroppers or malicious entities who want to read/modify this information. More importantly, blockchain transactions should not reveal any information about the sensed data.
- *Anonymity/unlinkability.* Neither the provider nor other users of the system should be able to learn anything about the identity of a user during the data reporting and payment phases of the protocol. Additionally, different sessions between the user and the provider should also be unlinkable to each other.
- *Protection against malicious providers.* Users should also be protected from providers who deny to pay (or pay less) for submitted data. Thus, it should not be possible for a provider to obtain the data for free or, more generally, in a price smaller than the one advertised by the rewarding function  $\mathcal{R}$ .
- *Protection against double redemption attacks and malicious user behavior.* Similarly, providers should be protected against malicious users who might (i) obtain a reward and refuse to release the promised data, (ii) release data whose value is less than the promised one, or (iii) try to obtain more than one rewards for the same data.

*Operational Assumptions:* We will be assuming that data submissions take place through *anonymous* communication channels that protect the identities of communicating devices.

In this work we don't consider *narrow tasking* attacks, where the requestor imposes strict limitations on attributes of sensing devices, or *selective tasking* attacks, where the task is pushed to a limited set of participants. These attacks reduce the size of the anonymity set thus making it easier for the collector to de-anonymize users [2]. These attacks are out of the scope of this work.

We also don't consider *data pollution* attacks, either. While we prevent malicious user behavior, we have no control over users that report *falsified* sensor data. One approach to this problem is attesting to the correct operation of the actual sensors as described in [26]. Another approach would be to use reputation frameworks that can penalize users that submit erroneous data [27], [28]. Such systems are complementary to our approach and can be used with our rewarding protocol. We note here that as the control of the sensing process is delegated to people, who may deliver (or not) sensed data or even choose to abuse the system, enforcing quality of data in mobile crowd-sensing is an important research area [29]. Here, we briefly describe how code attestation can be used in this respect. In particular, if user devices come equipped with a trusted platform module (TPM) chip, they can be used to vouch for the correct operation of the sensors involved. Furthermore, the TPM supports unlinkable keys that can be used to maintain anonymity between different parties who require the proof of identity. These keys can be used to digitally sign a hash of the sensed data thus ensuring not only the anonymity of the transaction but the truthfulness of the submission through appropriate code attestation.

Finally, while we make every effort to make our system privacy-preserving, we should note that Bitcoin is only pseudonymous by design; but pseudonymity is not enough to achieve privacy. As all blockchain transactions are publicly visible, if a bitcoin address is ever linked to a real-world identity (e.g. when physically buy something or having an item delivered home) then all past (and future) transactions can be linked to that identity. Thus, while our protocol ensures that *ephemeral* bitcoin addresses are used and data submissions and payments are unlinkable to each other, additional precautions are required to *spend* the bitcoins collected [30].

The desirable set of security properties expected from our system will be defined by means of experiments using a polynomial time adversary  $\mathcal{A}$  who may control other users or eavesdrop on honest ones. Additionally,  $\mathcal{A}$  has access and may manipulate the messages exchanged in the scope of the protocol.  $\mathcal{A}$ 's behavior will be captured by the set of oracles defined below:

- In  $\text{Register}^*(\mathcal{U})$ ,  $\mathcal{A}$  initiates **Register** with an honest  $\mathcal{P}$  assuming there are no pending  $\text{Register}^*$  calls for  $\mathcal{U}$  yet. We assume that the secret keys  $sk_{\mathcal{U}}$  and  $sk_{\mathcal{U}}^e$  are unique and unknown to the adversary.
- In  $\text{Submit}^*(\mathcal{U})$ ,  $\mathcal{A}$  initiates the **Submit** protocol with an honest  $\mathcal{P}$  for some data  $m_1, \dots, m_n$ .
- In  $\text{Verify}^*(val)$ ,  $\mathcal{A}$  initiates the **Verify** procedure with an honest  $\mathcal{P}$  for input  $val$ .

- In  $\text{Release}^*(\mathcal{U}, val)$ ,  $\mathcal{A}$  initiates the **Release** procedure with an honest  $\mathcal{P}$  and input  $val$ .

The properties of authorization, balance between value of data and payments, and double-redemption detection are formally defined below. The first property ensures that only authorized users can obtain rewards for data they submit.

*Definition 2 (Authorization):* A data submission and rewarding scheme protects against unauthorized data submissions and payments if for any PPT adversary  $\mathcal{A}$ , the advantage of  $\mathcal{A}$  in the experiment  $\text{Exp}_{\mathcal{A}}^{\text{Auth}}(\kappa)$  below is negligible.

#### Experiment $\text{Exp}_{\mathcal{A}}^{\text{Auth}}$

Let  $(CRS, (pk_{\mathcal{P}}, sk_{\mathcal{P}})) \leftarrow \text{Setup}(1^\kappa)$ . Then run  $\mathcal{A}^{\text{Register}^*, \text{Submit}^*, \text{Verify}^*, \text{Release}^*}(pk_{\mathcal{U}}^e, pk_{\mathcal{P}})$ . The experiment outputs 1 iff

- 1)  $\mathcal{A}$  holds a signed credential  $(pk_{\mathcal{U}}^e, sk_{\mathcal{U}}^e)$  that has not been the output of any previous  $\text{Register}^*$  query; or
- 2)  $\mathcal{A}$  successfully submits data using the  $\text{Submit}^*$ ,  $\text{Verify}^*$ ,  $\text{Release}^*$  oracles such that  $\mathcal{P}$  is convinced that the calls involve a valid credential  $pk_{\mathcal{U}}^e$  that has not been the output of a prior  $\text{Register}^*$  call..

The property of Double-redeeming Detection (DrD), ensures that data submissions initiated by some user for the same sensing task can be detected and prevented.

*Definition 3 (Double-Redeeming Detection):* A data submission and rewarding scheme successfully prevents double-redeeming attempts if for any PPT adversary  $\mathcal{A}$ , the advantage of  $\mathcal{A}$  in the experiment  $\text{Exp}_{\mathcal{A}}^{\text{DrD}}(\kappa)$  below is negligible.

#### Experiment $\text{Exp}_{\mathcal{A}}^{\text{DrD}}$

Let  $(CRS, (pk_{\mathcal{P}}, sk_{\mathcal{P}})) \leftarrow \text{Setup}(1^\kappa)$ . Then run  $\mathcal{A}^{\text{Register}^*, \text{Submit}^*, \text{Verify}^*, \text{Release}^*}(pk_{\mathcal{P}})$ . The experiment returns 1 iff  $\mathcal{A}$  makes two successful **Submit** or **Release** queries to the same data, and at least one of the following is true:

- The user public-keys extracted from the queries are  $pk_{\mathcal{U}}^1$  and  $pk_{\mathcal{U}}^2$ , with  $pk_{\mathcal{U}}^1 \neq pk_{\mathcal{U}}^2$  or
- The double-redeeming tags shown in these two queries are  $t_1$  and  $t_2$  respectively, with  $t_1 \neq t_2$
- **IdentDR** with these two transactions outputs 0.

We next consider the provider balance property (**BaP**). This property ensures that the amount redeemed for data  $m_1, \dots, m_n$  cannot exceed  $\mathcal{R}(m_1, \dots, m_n)$ , the value  $val$  returned by the application of the rewarding function  $\mathcal{R}$  on data  $m_i$ . This also suggests that an adversarial user cannot cheat an honest provider by getting a reward and releasing no data, or by releasing data of value less than  $val$ .

*Definition 4 (Provider Balance):* A data submission and rewarding scheme achieves Provider Balance if for any PPT adversary  $\mathcal{A}$ , the advantage of  $\mathcal{A}$  in the experiment  $\text{Exp}_{\mathcal{A}}^{\text{BaP}}(\kappa)$  below is negligible.

#### Experiment $\text{Exp}_{\mathcal{A}}^{\text{BaP}}$

Let  $(CRS, (pk_{\mathcal{P}}, sk_{\mathcal{P}})) \leftarrow \text{Setup}(1^\kappa)$ . Then run  $\mathcal{A}^{\text{Register}^*, \text{Submit}^*, \text{Verify}^*, \text{Release}^*}(pk_{\mathcal{P}})$ . The experiment outputs 1 if one of the following happens:

- 1)  $\mathcal{A}$  was able to extract a valid keypair  $(pk_{\mathcal{U}}^e, sk_{\mathcal{U}}^e)$  which can now use it to sign a new bitcoin transaction using  $sk_{\mathcal{U}}^e$ , thus spending another user's reward; or
- 2)  $\mathcal{A}$  obtains a payment larger than  $\mathcal{R}(m_1, \dots, m_n)$  for data submitted with  $pk_{\mathcal{U}}^e$ ; or
- 3)  $\mathcal{A}$  claims another payment for the same data and **IdentDR** outputs zero.

In an entirely analogous way, the user balance property (**BaU**) prevents a malicious provider  $\mathcal{A}$  to cheat an honest user by paying less than what the user deserves for the data provided.

*Definition 5 (User Balance):* A data submission and rewarding scheme achieves User Balance if for any PPT adversary  $\mathcal{A}$ , the advantage of  $\mathcal{A}$  in the experiment  $\text{Exp}_{\mathcal{A}}^{\text{BaU}}(\kappa)$  below is negligible.

#### Experiment $\text{Exp}_{\mathcal{A}}^{\text{BaU}}$

Let  $(CRS, (pk_{\mathcal{P}}, sk_{\mathcal{P}})) \leftarrow \text{Setup}(1^\kappa)$ . Then run  $\mathcal{A}^{\text{Register}^*, \text{Submit}^*, \text{Verify}^*, \text{Release}^*}(pk_{\mathcal{U}}^e, pk_{\mathcal{P}})$ . The experiment outputs 1 if one of the following happens:

- 1)  $\mathcal{A}$  was able to extract a valid keypair  $(pk_{\mathcal{U}}^e, sk_{\mathcal{U}}^e)$  which can now use it to sign a new bitcoin transaction using  $sk_{\mathcal{U}}^e$ , hence spending another user's reward; or
- 2)  $\mathcal{A}$  pays less than  $\mathcal{R}(m_1, \dots, m_n)$  for data submitted with  $pk_{\mathcal{U}}^e$ ; or  $\mathcal{A}$  makes a payment that is smaller than the value  $\mathcal{R}(m_1, \dots, m_n)$  of previously submitted data with  $pk_{\mathcal{U}}^e$ .
- 3)  $\mathcal{A}$  recovers the key  $K$  and decrypts the encrypted data  $C$ .

We next turn our attention to privacy. The goal of a malicious provider  $\mathcal{A}$  is to identify the user or link protocol operations. In general, data submissions and payments (even by the same user) should not be linkable to the user, and the actions of one user should not be distinguishable from the actions of another user. To formalize this behavior, we introduce an additional oracle **Corrupt**( $\mathcal{U}$ ) which allows  $\mathcal{A}$  obtain the user's secret key  $sk_{\mathcal{U}}$ .

*Definition 6 (Privacy):* A data submission and rewarding scheme is user-private if for any PPT adversary  $\mathcal{A}$ , the advantage of  $\mathcal{A}$  in the experiment  $\text{Exp}_{\mathcal{A}}^{\text{Priv}}(\kappa)$  below is negligibly close to  $1/2$ .

## IV. TOOLS

In what follows we describe the main tools we will be using in our proposal.

Experiment  $\text{Exp}_A^{\text{Priv}}$ 

Let  $(CRS, (pk_P, sk_P)) \leftarrow \text{Setup}(1^\kappa)$ . The experiment consists of the following phases:

- *Learning phase*:  $\mathcal{A}$  asks any number of users to Register\*, Submit\*, Release\* data multiple times.  $\mathcal{A}$  may also Corrupt any number of users except two that will be used in the challenge phase.
- *Challenge phase*:  $\mathcal{A}$  picks two honest users  $U_1$  and  $U_2$  whose key material has not been corrupted in the previous phase. The challenge oracle selects at random a bit  $b \in \{0, 1\}$  and sets user  $U_L$  equal to  $U_b$  and  $U_R$  equal to  $U_{1-b}$ .  $\mathcal{A}$  chooses data  $m_1, \dots, m_n$  and may execute the following steps:
  - Ask users  $U_L$  or  $U_R$  to submit data using Submit\*.
  - Ask users  $U_L$  or  $U_R$  to receive payments using Release\*.
- *Post-Challenge phase*:  $\mathcal{A}$  asks both users to further Submit\*, Release\* data multiple times and concurrently.

The experiment returns 1 if  $\mathcal{A}$  outputs a guess bit  $b'$  which is equal to  $b$ .

## A. ZKSNARKS

Our protocol is based on the security of zero-knowledge Succinct Non-interactive ARguments of Knowledge (*zkSNARKs*) as developed in [31]. *zkSNARKs* allow a prover to convince a verifier about the validity of an NP statement by constructing a small size proof  $\pi$ .

*zkSNARKs* consist of three algorithms setup, prove and verify. The *setup* algorithm, given a security parameter  $\kappa$  and an NP language  $L$  corresponding to a relation  $R = \{x, w\}$  where  $w$  is a witness that  $x \in L$ , outputs a common reference string  $CRS$  consisting of a public evaluation key for proving that  $w$  is a witness for  $x$  and a public verification key used in the verification of such statements. *Prove* is an algorithm that, given  $CRS$ ,  $x$  and  $w$ , produces a proof  $\pi$  that  $w$  is a valid witness for  $x$ . *Verify* is an algorithm that given  $CRS$ ,  $x$ ,  $\pi$  outputs either 'Accept' or 'Reject' depending on the validity of the proof.

The following properties are expected by a secure *zkSNARK* scheme [31]:

- *Completeness*. For  $(x, w) \in R$ , a proof  $\pi$  can be generated such that  $\text{Verify}(x, \pi)$  outputs 'Accept'.
- *Soundness*. No malicious prover can produce a proof  $\pi$  for a false statement such that  $\text{Verify}(x, \pi)$  outputs 'Accept'.
- *Zero-knowledge*. A polynomial time simulator  $S$  exists such that for any  $x \in L_Q$ ,  $S$  can generate a proof that cannot be distinguished from a real one.

*Remark 1*: The common reference string ( $CRS$ ) in *zkSNARKs* must be generated in a trustworthy manner. Otherwise, a malicious verifier (in this case the service provider)

can construct a  $CRS$  that allows it to derive the user's secret keys or data. This can be prevented if the user examines the well-formedness of the  $CRS$ . Thus no trust is really placed on the service provider. Another possibility is to use Subversion-NIZK [32] proofs, where the zero-knowledge property is maintained even when a (possibly malicious) verifier chooses the  $CRS$ .

## B. BLOCKCHAIN

A blockchain is decentralized and distributed public ledger that is used to record transactions. It is a linked-list data structure consisting of block of transactions, where each block also contains the hash of the previous listed block. Thus attempting to modify an existing transaction will result in a chain of updates; however this cannot happen without consensus by the majority of the peers maintaining the ledger. This property gives the blockchain its immutable, verifiable and transparent character. In this work, we will consider both simple bitcoin payments and smart contracts to handle user rewards.

## 1) BITCOIN BLOCKCHAIN

The blockchain will be used as the means to reward users for the sensed data they provide. Payments are made and sent to *bitcoin addresses* which are created privately by users of the system. Addresses are bound to (hashes of) user public keys and transactions must be signed to be considered authentic. Hence addresses act as user pseudonyms. However, as anyone can see the balance and transactions on these addresses, it is advised that each such address is only used once.

Transactions can be used to transfer bitcoins from one party to another. One such example is the *Pay-to-Script-Hash* (P2SH) [33] in which a payment is made to a script matching a particular hash value. In our case, the hash value will correspond to the hash of the key encrypting the sensed data, and the user will have to reveal this key in order to get paid. The encrypted data and the key will be sent from  $U$  to  $\mathcal{P}$  *offchain* in order for the P2SH to be constructed by  $\mathcal{P}$  and posted to the blockchain. This P2SH transaction can also be made a *time-locked* one in which case the provider can re-claim its money if the user misbehaves and does not release the key before a specific time bound.

## 2) ETHEREUM BLOCKCHAIN

While the bitcoin blockchain can handle only relatively simple scripts, an alternative is to have computer code stored and executed by all nodes in the blockchain network. Such code is also known as a smart contract. Smart contracts are called by addressing a transaction to them and miners are responsible for processing them. The most well-known smart contract enabled blockchain is Ethereum [34]. In Ethereum, miners and rewarded for processing transactions via fees which are paid in gas. These fees depend on the complexity of the smart contract scripts.

In the experimental section, we will demonstrate that only very simple scripts and bitcoin transactions are needed to

handle user rewards. This is due to the fact that the blockchain network is only used to post the ‘key-for-payment’ transactions. The rest of the protocol steps (can be combined to a *single* message) are taking place *offchain*, thus making the entire protocol very efficient.

## V. PROTOCOL DETAILS

### A. OVERVIEW

In the main operational phase of our protocol, a user  $\mathcal{U}$  wants to receive a payment from a service provider  $\mathcal{P}$  for data  $m_1, \dots, m_n$  sensed with her smart device. The data worths some value  $val$  which can be computed by the application of a rewarding function  $\mathcal{R}()$ .  $\mathcal{R}()$  is announced by the provider and its purpose is to capture the value of the data for attributes set forth by  $\mathcal{P}$  for the given task such as location, sensing time, sampling frequency, type of sensor used, and so on.

The utility  $val = \mathcal{R}(m_1, \dots, m_n)$  is calculated locally at the mobile device, as all necessary information is already available to the user. However, the data cannot be released to the provider as  $\mathcal{P}$  may act maliciously and refuse to pay the user. Hence the data is sent encrypted using a one-time key  $K$ . This key will be released only when the provider posts a time-locked transaction offering  $val$  bitcoins in exchange for  $K$ . However, before the provider posts this transaction, it must be convinced that the encrypted data worth value  $val$  and have not been submitted before by  $\mathcal{U}$ . This is an NP statement and can be implemented efficiently using a *zkSNARK* proof  $\pi$ . Once this proof is verified, the key-for-money transaction takes place.

This ensures that no party can cheat; the provider is certain that whoever claims the money must present the correct decryption key  $K$  used in the proof  $\pi$ . Similarly, the user is sure that she will get her money when she posts  $K$ . The above approach ensures that either both parties will get what they deserve or none can be in disadvantage.

A final issue that needs to be taken care is that the user’s transaction must be signed with her secret key to be valid. However to ensure unlinkability and prevent  $\mathcal{P}$  associating the signing key with multiple submissions of data from  $\mathcal{U}$ , this key has to be an *one-time, ephemeral* key. But while this ephemeral key will not be tied to a particular user, we need to ensure that only authorized users can submit data. This tension between unlinkability and accountability will be the focus of subsequent sections.

### B. SETUP

$\text{Setup}(1^\kappa)$  is used to initialize the system. This method creates the  $\mathcal{CRS}$  used in the *zkSNARKs* operations. The provider’s public/private key  $(pk_{\mathcal{P}}, sk_{\mathcal{P}})$  is generated as well.

### C. REGISTRATION

One of the key requirements of our protocol is that a user cannot claim more than one reward for the data they provide (double-redemption also known as double-spending). A simple way to prevent this is to embed information in the submitted data so that the user is either prevented or identified

if she tries to double-redeem. This information will have the form of a *redeeming tag*  $\tau$  that will be *unique* for the sensing task the user is responding to. Thus, if a user attempts to reap multiple rewards for a given task, she will be prevented from doing so. However, this must be carefully done to ensure that the user remains anonymous when she follows the protocol.

To this respect, before a user is allowed to participate in the crowd-sourcing application, she must register first. Thus she generates a public-private key pair  $(pk_{\mathcal{U}}, sk_{\mathcal{U}})$  and registers  $pk_{\mathcal{U}}$  with the Service Provider  $\mathcal{P}$ .  $\mathcal{P}$  signs the user’s public key and produces a certificate  $cert_{\mathcal{U}}$  for the authenticity of  $pk_{\mathcal{U}}$ . The provider ensures that  $pk_{\mathcal{U}}$  is unique and stores it in its database along with any other useful information about the user. We define operation  $\text{CertVerify}_{\mathcal{P}}(cert_{\mathcal{U}})$  that returns 1 if the public key of  $\mathcal{U}$  is indeed signed by  $\mathcal{P}$ . This can be used to test the authenticity of the public key contained in  $cert_{\mathcal{U}}$ .

It is important to note here that this registration step correctly binds each user identity to a *unique* credential. This is to ensure accountability by preventing Sybil attacks, guaranteeing that only authenticated users participate in the system, and eventually, preventing double-redemptions of data. However, as we will see later on this step does not have any effects on user privacy since we make sure that users remain anonymous in all subsequent phases of the protocol. The user’s key pair can now be used to establish an *ephemeral* bitcoin address that will be used in the actual rewarding phase to provide for unlinkability between the reported data and the rewards.

In view of this,  $\mathcal{U}$  generates a new public-private key pair  $(pk_{\mathcal{U}}^e, sk_{\mathcal{U}}^e)$  and asks the provider to *blindly* sign  $pk_{\mathcal{U}}^e$  using any secure blind signature scheme. For example, if the provider possesses an RSA key pair  $(e_{\mathcal{P}}, d_{\mathcal{P}})$ , the user can first send  $r^{e_{\mathcal{P}}}H(pk_{\mathcal{U}}^e)$  to the provider, where  $r$  is some blinding factor chosen by the user and  $H$  a secure hash function. After signing with  $\mathcal{P}$ ’s private key, the user obtains a signature  $r(H(pk_{\mathcal{U}}^e))^{d_{\mathcal{P}}}$ , which after removal of  $r$ , is a signature on the hash of the ephemeral key  $pk_{\mathcal{U}}^e$ . Thus, when the provider sees such a bitcoin address, it knows it is coming from an authenticated user but cannot tell which user it is due to the security of the blind signature scheme. When all these steps are performed, the user is consider authorized and can participate in the crowd-sensing task.

It is necessary to use *different* ephemeral keys to collect rewards in order to provide unlinkability between these rewards. Thus each bitcoin address should only be used *once* (recall Section III-B), hence whenever the user wants to submit data for a new task, she has to obtain a new bitcoin address. The user may have as many ephemeral keys authenticated as she likes simply by asking the provider to blindly sign multiple keys as described above. However, if the provider does not want to do so, new ephemeral keys can be generated during data submission. Section V-E1 describes how this can be done in an unlinkable way.

### D. TASK ADVERTISING

When the provider needs to collect data about a particular sensing task, it has to advertise the task to the users.

As mentioned in Section III, this is typically the job of a task server which users may contact. Here, however, we will not distinguish between the service provider and the task server.

Each sensing task may ask for user data or other useful information based on various criteria (region, sampling frequency, etc.). Tasks can be either downloaded by users (*pull* model) or sent to them when the provider has a new sensing task (*push* model). The tasking and downloading processes may endanger the privacy of the participants in several ways (recall the *narrow and selective tasking* attacks mentioned in Section III-B. These attacks are out of the scope of this work as our main focus is on rewarding, however we must insist that users communicate with the service provider through *anonymizing* networks like TOR.

An example of a task published by  $\mathcal{P}$  is shown below. In this task participants have to report 5 min temperature readings in London for a duration of one day.

$$s = ( \text{taskID} = \#53621, \\ \text{Location} = \text{London}, \\ \text{sensingType} = \text{getTemperature}, \\ \text{Frequency} = 5 \text{ min}, \\ \text{Start} = \text{April 1, 2022}, \\ \text{Duration} = 24 \text{ h} )$$

Such a task will be unique as indicated by its *taskID* and must be *signed* by the provider to be valid. Thus any user downloading such a task will know that is an authentic one. The uniqueness of the task will come into play later on as it will be crucial to ensure that no user can double-redeem, i.e. obtain more than one reward for the same task.

In addition to the task, the provider will publish its rewarding function  $\mathcal{R}()$ . Users can use this function to compute the amount of reward they will get for the data they provide. The reward *val* for data  $m_1, \dots, m_n$  will result from the application of this function on the data and some auxiliary variables  $a_i$ , i.e.  $val = \mathcal{R}(a_1, \dots, a_k, m_1, \dots, m_n)$ . The rewarding function is also public information and is known to the users.

### E. RESPONDING TO A SENSING TASK

Let  $m_1, \dots, m_n$  be the measurements to be reported to  $\mathcal{P}$ . User  $\mathcal{U}$  applies the rewarding function to compute the value  $val = \mathcal{R}(m_1, \dots, m_n)$  of the data. Now  $\mathcal{U}$  has to convince  $\mathcal{P}$  about the utility of the data, however without sending the data as is. To do so,  $\mathcal{U}$  engages in the following steps:

- 1)  $\mathcal{U}$  encrypts the data with a one-time symmetric key  $K$  to produce a ciphertext  $C = E_K(m_1, \dots, m_n)$ . Both  $C$  and the hash  $h_K$  of  $K$  will be sent to  $\mathcal{P}$  along with a proof  $\pi$  that proves knowledge of  $K$  and the valuation *val* of the data. In addition to  $C$ , the user constructs and sends a double-redeeming tag  $\tau = H(\text{taskID}, sk_{\mathcal{U}})$ , where  $sk_{\mathcal{U}}$  denotes the *long term* private key of  $\mathcal{U}$ . The role of  $\tau$  is to prevent the user from redeeming the same data twice for the given task.

- 2) Given public information  $(\text{taskID}, C, h_K, val, \tau)$ ,  $\mathcal{U}$  constructs a *zkSNARK* proof  $\pi$  that demonstrates it knows  $(m_1, \dots, m_n, K, cert_{\mathcal{U}}, sk_{\mathcal{U}})$  such that:

- a)  $K$  was used for the encryption of the data and  $H(K) = h_K$
- b)  $\tau = H(\text{taskID}, sk_{\mathcal{U}})$
- c)  $cert_{\mathcal{U}}$  is a valid certificate signed by  $\mathcal{P}$
- d)  $val = \mathcal{R}(m_1, \dots, m_n)$ .

More formally, if  $\mathcal{R}$  is the rewarding function, we define the NP language  $\mathcal{L}_{\mathcal{R}}$  for the *zkSNARK*-proof system to be the set of the following NP statements:

$$\mathcal{L}_{\mathcal{R}} = \left\{ (\text{taskID}, C, h_K, \tau, val) \mid \begin{array}{l} \exists \{m_i\}_{i=1}^n, K, cert_{\mathcal{U}}, sk_{\mathcal{U}} : \\ h_K = H(K) \\ C = E_K(m_1, \dots, m_n) \\ \tau = H(\text{taskID}, sk_{\mathcal{U}}) \\ \text{CertVerify}_{\mathcal{P}}(cert_{\mathcal{U}}) \\ \text{GenVerify}(pk_{\mathcal{U}}, sk_{\mathcal{U}}) = 1 \\ val = \mathcal{R}(m_1, m_2, \dots, m_n) \end{array} \right\},$$

where operation  $\text{GenVerify}(pk_{\mathcal{U}}, sk_{\mathcal{U}})$  returns 1 if  $(pk_{\mathcal{U}}, sk_{\mathcal{U}})$  is a valid public-private key pair. This is needed to ensure that the user created the redeeming tag with its long term key, thus linking  $sk_{\mathcal{U}}$  with the public key in  $cert_{\mathcal{U}}$ . For example, if  $x$  is the user's secret key and  $y = g^x$  her public key for some generator  $g$ , then  $\text{GenVerify}$  simply checks that  $y = g^x$  and returns 1 if the test succeeds.

At this point,  $\mathcal{U}$  sends the following message to  $\mathcal{P}$ .

$$\mathcal{U} \xrightarrow[\text{offchain}]{\text{Anon}} \mathcal{P} : \text{Enc}_{\mathcal{P}}(C), h_K, val, \tau, \pi, pk_{\mathcal{U}}^e, \text{Sig}_{\mathcal{P}}(pk_{\mathcal{U}}^e), \sigma_e \quad (1)$$

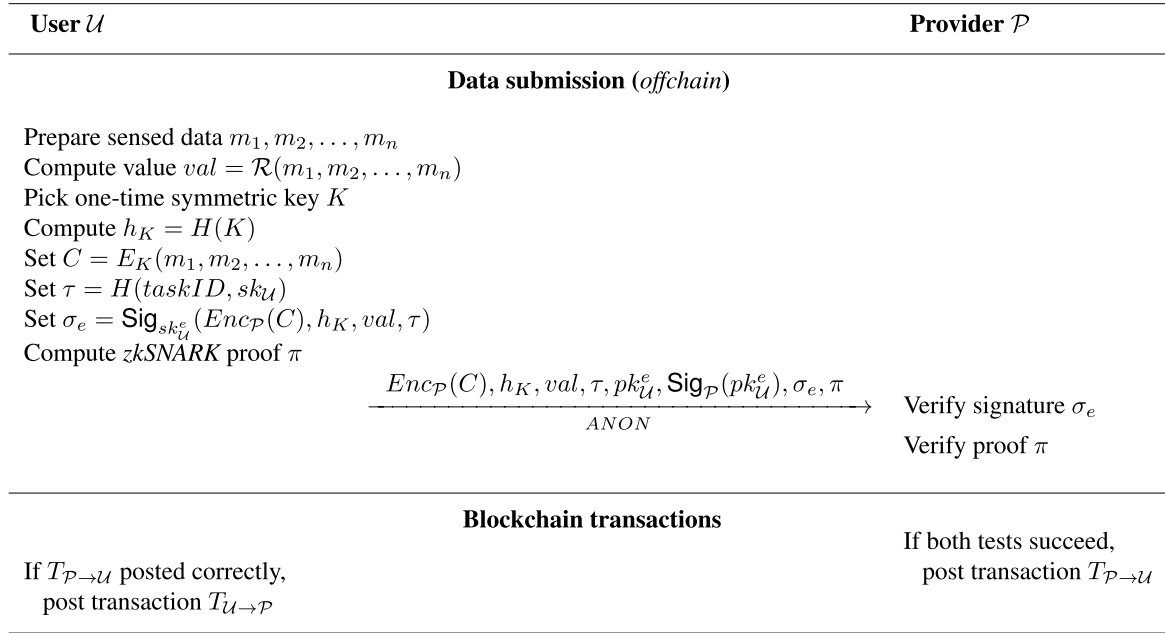
This message is signed with  $\mathcal{U}$ 's *ephemeral* key  $sk_{\mathcal{U}}^e$  to produce a signature  $\sigma_e$  and is sent through an anonymizing network connection. The ciphertext  $C$  is *encrypted* with  $\mathcal{P}$ 's public key to prevent third parties from accessing the data in  $C$  once the key  $K$  is posted to the blockchain by  $\mathcal{U}$ . Thus only  $\mathcal{P}$  can obtain the data. This message takes place *offchain*.

Notice that  $\mathcal{U}$  also sends its ephemeral public key  $pk_{\mathcal{U}}^e$  that was blindly signed during the registration phase.  $\text{Sig}_{\mathcal{P}}(pk_{\mathcal{U}}^e)$  is the associated signature that can be used to verify the validity of the key. This ephemeral key (bitcoin address) will be used by the provider to pay for the data during the onchain phase of the protocol.

### 1) GENERATING EPHEMERAL KEYS ON THE FLY

To ensure unlinkability, new ephemeral keys must be generated with every new data submission and payment made. To do this, the process can be combined with data submission, hence there is no need to run the registration procedure from scratch.

The user creates a new ephemeral key  $pk_{\mathcal{U}}^e$ , and sends a blinded version  $r^{e_{\mathcal{P}}} h_{e'}$  of its hash  $h_{e'}$  along with the Message showed in (1). The provider authenticates the user as the message is signed using the *current* ephemeral key. Then proceeds to sign the blinded hash value. Notice that  $\mathcal{P}$  cannot



**FIGURE 2.** Data submission and rewarding. The offchain part consists of just one message from  $\mathcal{U}$  to  $\mathcal{P}$ .

link the two keys together due to the security of the blind signature used. Hence new, signed ephemeral keys can be generated that cannot be linked to each other.

### F. GETTING PAID FOR THE DATA

Once  $\mathcal{P}$  obtains Message 1, it first verifies the signature  $\sigma_e$  as coming from an authenticated ephemeral key  $pk_{\mathcal{U}}$  (recall the blind signature during registration) by checking  $\text{Sig}_{\mathcal{P}}(pk_{\mathcal{U}}^e)$ . Then it verifies the correctness of the proof  $\pi$ .

If everything checks out, it posts to the blockchain a time-locked transaction  $T_{\mathcal{P} \rightarrow \mathcal{U}}$  which says that  $\mathcal{P}$  offers  $val$  bitcoins to  $\mathcal{U}$  under the condition that “ $\mathcal{U}$  must present a pre-image  $K$  to the hash value  $h_K$  as well as a signature within some time window  $t$ ”; if the conditions are not satisfied the bitcoins return to  $\mathcal{P}$ . More precisely, the transaction  $T_{\mathcal{P} \rightarrow \mathcal{U}}$  has an output of  $val$  bitcoins that can be redeemed by a (future) transaction  $T$  if one of the following is true:

- 1)  $T$  is signed by  $\mathcal{U}$  and contains a valid pre-image of  $h_K$ ,  
or
- 2)  $T$  is signed by  $\mathcal{P}$  and the time window  $t$  has expired.

The transaction  $T_{\mathcal{P} \rightarrow \mathcal{U}}$  is satisfied if  $\mathcal{U}$  posts a transaction  $T_{\mathcal{U} \rightarrow \mathcal{P}}$  that contains  $K$ . This would satisfy condition 1 of  $T_{\mathcal{P} \rightarrow \mathcal{U}}$  and so  $val$  bitcoins are transferred to  $\mathcal{U}$ . If  $\mathcal{U}$  does not act within the time window  $t$ , then  $\mathcal{P}$  can sign and post a transaction  $T$  that returns the  $val$  bitcoins back to him, thus satisfying the second condition of  $T_{\mathcal{P} \rightarrow \mathcal{U}}$ . If  $T_{\mathcal{P} \rightarrow \mathcal{U}}$  is posted with the wrong amount,  $\mathcal{U}$  simply aborts and does not release  $K$ .

This concludes the *onchain* phase and the description of the protocol. A summary of the protocol steps is shown in Figure 2. Contrary to prior works, a single *offchain* message

is sent from  $\mathcal{U}$  to  $\mathcal{P}$ , thus contributing to the efficiency of the protocol.

*Remark 1:* To guarantee atomicity of transactions and ensure that a curious provider does not learn  $K$  without paying,  $\mathcal{U}$  must submit  $T_{\mathcal{U} \rightarrow \mathcal{P}}$  well before the expiration of  $t$ . To see why, consider the case where  $\mathcal{U}$  posts  $T_{\mathcal{U} \rightarrow \mathcal{P}}$  just before the eclipse of the time window  $t$  while  $\mathcal{P}$  posts  $T$  immediately after  $t$  has expired. Then, it is possible (due to network delays or miners choice) that  $T$  ends up in the blockchain,  $T_{\mathcal{U} \rightarrow \mathcal{P}}$  is dropped and as a result  $\mathcal{U}$  cannot receive the payment. However, since  $\mathcal{U}$  posted  $T_{\mathcal{U} \rightarrow \mathcal{P}}$  in the blockchain network,  $\mathcal{P}$  can peek at  $K$  even without paying. To prevent this, we need to consider the time it takes for a transaction to be permanently saved in the blockchain. In particular, denote by  $t_{\delta}$  the time required for a transaction to be validated and stored in a blockchain block. Typically, this requires 6 confirmations (the more confirmations a transaction gets, the more probable it is that it will end up in the long-term consensus chain) which corresponds to about 60 minutes in Bitcoin and about 2 minutes in the Ethereum blockchain [35]. Thus,  $\mathcal{U}$  should submit  $T_{\mathcal{U} \rightarrow \mathcal{P}}$  before  $t - t_{\delta}$  to ensure her transaction is confirmed in the blockchain with high probability.

## VI. SECURITY ANALYSIS

In this section, we analyze the security and privacy properties expected from our system.

### A. AUTHORIZATION

By Definition 2,  $\mathcal{A}$  wins the authorization game if  $\mathcal{A}$  is in possession of a valid ephemeral key that is not the result of any **Register\*** query. If this is possible, it means that  $\mathcal{A}$  either

knows or can forge the key of  $\mathcal{U}$  without  $\mathcal{P}$ 's involvement. Both conditions contradict the security of the underlying signature schemes used.

Alternatively,  $\mathcal{A}$  can make successful calls to **Submit\***, **Verify\***, and **Release\*** queries such that  $\mathcal{P}$  believes that these calls involve a valid ephemeral key that is not the result of **Register\*** up to this moment. But this is not possible since  $\mathcal{A}$  could not have generated the signature without knowledge of the secret ephemeral key. The security of the signing algorithms ensures that this cannot happen.

The above intuition is captured by the following theorem:

*Theorem 1 (Authorization):* Assume the signature scheme used in the creation of the authorization credential is unforgeable. Then our rewarding scheme satisfies Definition 2.

*Proof:* Consider adversary  $\mathcal{A}$  in the experiment  $\text{Exp}_{\mathcal{A}}^{\text{Auth}}$ . Then

$$\text{Adv}_{\mathcal{A}}^{\text{Auth}}(\kappa) \leq \text{Adv}_{\text{SIG},\mathcal{B}}^{\text{UF-CMA}}(\kappa),$$

where  $\text{Adv}_{\text{SIG},\mathcal{B}}^{\text{UF-CMA}}(\kappa)$  is the advantage of an adversary  $\mathcal{B}$  that breaks the unforgeability of the signature scheme.

We will show that if either of the cases mentioned above happens, an adversary  $\mathcal{B}$  exists that succeeds in computing a signature using knowledge of the public key only. In order to answer the challenge in the signature unforgeability game,  $\mathcal{B}$  simulates **Register** queries by first generating an ephemeral key pair  $(pk_{\mathcal{U}}^e, sk_{\mathcal{U}}^e)$  then calling  $\mathcal{A}^{\text{Register*}}$ .

If  $\mathcal{A}$  forges a signature on the hash of  $pk_{\mathcal{U}}^e$  (case 1 above) then  $\mathcal{B}$  could use this to win the unforgeability game. Alternatively, if  $\mathcal{A}$  manages to pass verification when executing **Submit\***, **Verify\***, and **Release\*** queries such that these calls involve a valid ephemeral key that has not been the output of an **Register** call (case 2) then again  $\mathcal{B}$  could use this signature as its forgery. But since the signature scheme is secure,  $\Pr[\mathcal{B} \text{ wins}]$  is bounded by  $\text{Adv}_{\text{SIG},\mathcal{B}}^{\text{UF-CMA}}(\kappa)$ . Thus,

$$\text{Adv}_{\mathcal{A}}^{\text{Auth}}(\kappa) \leq \text{Adv}_{\text{SIG},\mathcal{B}}^{\text{UF-CMA}}(\kappa)$$

■

## 1) DOUBLE-REDEEMING DETECTION

Each double-redeeming tag  $\tau = H(\text{taskID}, sk_{\mathcal{U}})$  is bound to a unique user identity (as is expressed by  $sk_{\mathcal{U}}$ ). Thus each user can only redeem once, as the data is associated with a specific task and  $sk_{\mathcal{U}}$ . Intuitively, a user can resubmit the same data only if it recovers the secret key  $sk_{\mathcal{U}'}$  of another user  $\mathcal{U}'$ . However, this is prevented by the one-wayness of the hash function and the zero knowledge property of the  $zkSNARK$  proof. Alternatively, a malicious user may try to provide a proof of an invalid statement to fool the verifier about the value of a redeeming tag.

The above intuition is captured by the following theorem:

*Theorem 2 (Double-Redeeming Detection):* If the hash function is one way, and the  $zkSNARK$  is sound and zero-knowledge, our rewarding scheme defends against

double-redeeming detection attempts as captured in Definition 3.

*Proof:* Consider adversary  $\mathcal{A}$  in the experiment  $\text{Exp}_{\mathcal{A}}^{\text{DrD}}$ . Then

$$\begin{aligned} \text{Adv}_{\mathcal{A}}^{\text{DrD}}(\kappa) &\leq q \cdot \text{Adv}_{\mathcal{B}}^{\text{OW}}(\kappa) \\ &\quad + q \cdot \text{Adv}_{\text{POK},\mathcal{C}}^{\text{ZK}}(\kappa) \\ &\quad + \text{Adv}_{\text{POK},\mathcal{C}}^{\text{sound}}(\kappa), \end{aligned}$$

where  $\text{Adv}_{\mathcal{B}}^{\text{OW}}(\kappa)$  is the advantage of an adversary  $\mathcal{B}$  that succeeds in breaking the one-way property of the hash function, and  $\text{Adv}_{\text{POK},\mathcal{C}}^{\text{ZK}}(\kappa)$ ,  $\text{Adv}_{\text{POK},\mathcal{C}}^{\text{sound}}(\lambda)$  denote the advantages of adversaries  $\mathcal{C}$ ,  $\mathcal{D}$  that succeed in breaking the soundness and zero-knowledge properties of the underlying  $zkSNARK$  scheme.

We start with the case (denote it by  $\text{C}_{\mathcal{A},1}^{\text{DrD}}$ ) where adversary  $\mathcal{A}$  observes a double-redeeming tag  $\tau = H(\text{taskID}, sk_{\mathcal{U}})$  and recovers  $sk_{\mathcal{U}}$ . Now she can act on behalf of  $\mathcal{U}$  and submit the same data under  $\mathcal{U}$ 's key. If this is possible, an adversary  $\mathcal{B}$  exists that attacks the one-wayness property of the hash function.  $\mathcal{B}$  starts by executing  $\mathcal{A}^{\text{Register*}, \text{Submit*}, \text{Release*}}()$  while simulating  $\mathcal{A}$ 's queries. As the attack occurs,  $\mathcal{A}$  recovers the hidden key  $sk_{\mathcal{U}}$  of one of the users. Since it is not known which of the polynomially many (say  $q$ ) tags causes this,  $\mathcal{B}$  selects one of these tags to answer the challenge. Therefore,  $\Pr[\mathcal{B} \text{ wins}] \geq \Pr[\text{C}_{\mathcal{A},1}^{\text{DrD}}]/q$ . However, since the hash function is one-way, we have that  $\Pr[\mathcal{B} \text{ wins}] \leq \text{Adv}_{\mathcal{B}}^{\text{OW}}(\kappa)$ , hence  $\Pr[\text{C}_{\mathcal{A},1}^{\text{DrD}}] \leq q \cdot \text{Adv}_{\mathcal{B}}^{\text{OW}}(\kappa)$ .

Now, consider the case (denote it by  $\text{C}_{\mathcal{A},2}^{\text{DrD}}$ ) where the adversary observes the proof  $\pi$  and extracts  $sk_{\mathcal{U}}$ . If this is possible, an adversary  $\mathcal{C}$  exists that uses  $\mathcal{A}$  to attack the zero-knowledge of the  $zkSNARK$  scheme. As before,  $\mathcal{C}$  starts by executing  $\mathcal{A}^{\text{Register*}, \text{Submit*}, \text{Release*}}()$  while simulating  $\mathcal{A}$ 's queries. Because event  $\text{C}_{\mathcal{A},2}^{\text{DrD}}$  occurs,  $\mathcal{C}$  would form a double-redeeming tag and submit it to its challenger in the ZK game. Let  $\pi$  be the proof it receives back. This is forwarded to  $\mathcal{A}$  to distinguish between the real and the simulated proof. Thus,  $\Pr[\mathcal{C} \text{ wins}] \geq \Pr[\text{C}_{\mathcal{A},2}^{\text{DrD}}]/q$ . However, since the  $zkSNARK$  is zero-knowledge,  $\Pr[\mathcal{C} \text{ wins}] \leq \text{Adv}_{\text{POK},\mathcal{C}}^{\text{ZK}}(\kappa)$ , hence  $\Pr[\text{C}_{\mathcal{A},2}^{\text{DrD}}] \leq q \cdot \text{Adv}_{\text{POK},\mathcal{C}}^{\text{ZK}}(\kappa)$ .

Finally, consider the case (denote it by  $\text{C}_{\mathcal{A},3}^{\text{DrD}}$ ), where  $\mathcal{A}$  successfully submits an invalid tag. If this is possible, an adversary  $\mathcal{D}$  exists that uses  $\mathcal{A}$  to attack the soundness of the  $zkSNARK$  proof system.  $\mathcal{D}$  executes  $\mathcal{A}^{\text{Register*}, \text{Submit*}, \text{Release*}}()$  while simulating  $\mathcal{A}$ 's queries. Because event  $\text{C}_{\mathcal{A},3}^{\text{DrD}}$  occurs,  $\mathcal{A}$  outputs a false proof  $\pi^*$  that fools a verifier.  $\mathcal{D}$  then forwards this proof to its challenger to attack the soundness of the scheme. Therefore,  $\Pr[\mathcal{D} \text{ wins}] \geq \Pr[\text{C}_{\mathcal{A},3}^{\text{DrD}}]$ . However, since the  $zkSNARK$  is sound, we have that  $\Pr[\text{C}_{\mathcal{A},3}^{\text{DrD}}] \leq \Pr[\mathcal{B} \text{ wins}] \leq \text{Adv}_{\text{POK},\mathcal{C}}^{\text{sound}}(\kappa)$ .

The proof of the theorem follows since

$$\text{Adv}_{\mathcal{A}}^{\text{Abr}}(\kappa) \leq \Pr[\text{C}_{\mathcal{A},1}^{\text{DrD}}] + \Pr[\text{C}_{\mathcal{A},2}^{\text{DrD}}] + \Pr[\text{C}_{\mathcal{A},3}^{\text{DrD}}].$$

■

## 2) PROVIDER BALANCE

Since both the authorization and double-redeeming properties hold, to win the provider balance game an adversarial user  $\mathcal{A}$  must extract a valid signing ephemeral key  $sk_{\mathcal{U}}^e$  and sign a new bitcoin transaction, sending the money to a bitcoin address owned by  $\mathcal{A}$ . This can only happen if forging signatures is possible. Similarly, obtaining a larger reward than the one embedded in the  $zkSNARK$  through the computation of  $\mathcal{R}()$  is also infeasible as this would break the soundness property of the proof system. Thus in both cases provider balance is preserved.

The above intuition is captured by the following theorem:

*Theorem 3 (Provider Balance):* If Theorem 1 is true, signatures are unforgeable and the  $zkSNARK$  is sound, our rewarding scheme satisfies Definition 4.

*Proof:* Consider adversary  $\mathcal{A}$  in the experiment  $\text{Exp}_{\mathcal{A}}^{\text{BaP}}$ . Then

$$\text{Adv}_{\mathcal{A}}^{\text{BaP}}(\kappa) \leq \text{Adv}_{\text{SIG}, \mathcal{B}}^{\text{UF-CMA}}(\kappa) + \text{Adv}_{\text{POK}, \mathcal{C}}^{\text{sound}}(\kappa),$$

where  $\text{Adv}_{\text{SIG}, \mathcal{B}}^{\text{UF-CMA}}(\kappa)$ ,  $\text{Adv}_{\text{POK}, \mathcal{C}}^{\text{sound}}(\kappa)$  are the advantages of adversaries  $\mathcal{B}$ ,  $\mathcal{C}$  that succeed in breaking the unforgeability of signature and the soundness of the underlying  $zkSNARK$  scheme, respectively.

We start with the case (denote it by  $\mathcal{C}_{\mathcal{A},1}^{\text{BaP}}$ ) where adversary  $\mathcal{A}$  observes a bitcoin transaction signed with  $sk_{\mathcal{U}}^e$  and extracts the signing key. If this is possible, an adversary  $\mathcal{B}$  exists that wins in the signature unforgeability game as follows:  $\mathcal{B}$  would simulate  $\mathcal{A}$ 's actions and wait until  $\mathcal{A}$  extracted the signing ephemeral key and posted a new bitcoin transaction with  $sk_{\mathcal{U}}^e$ . Then  $\mathcal{B}$  could win the signature unforgeability game by submitting this signature to its challenger. Hence  $\Pr[\mathcal{B} \text{ wins}] \geq \Pr[\mathcal{C}_{\mathcal{A},1}^{\text{BaP}}]$ . However, as the signature scheme is unforgeable, we have that  $\Pr[\mathcal{B} \text{ wins}] \leq \text{Adv}_{\text{SIG}, \mathcal{B}}^{\text{UF-CMA}}(\kappa)$ , hence  $\Pr[\mathcal{C}_{\mathcal{A},1}^{\text{BaP}}] \leq \text{Adv}_{\text{SIG}, \mathcal{B}}^{\text{UF-CMA}}(\kappa)$ .

Now consider the case (denote it by  $\mathcal{C}_{\mathcal{A},2}^{\text{BaP}}$ ), where  $\mathcal{A}$  successfully submits a false proof  $\pi^*$ . If this is possible, an adversary  $\mathcal{C}$  exists that breaks the soundness of the underlying  $zkSNARK$  scheme as follows.  $\mathcal{C}$  starts by executing  $\mathcal{A}^{\text{Register}^*, \text{Submit}^*, \text{Release}^*}()$  while simulating  $\mathcal{A}$ 's queries. When  $\mathcal{A}$  produces the false proof  $\pi^*$ ,  $\mathcal{C}$  would submit this proof to its challenger in the ZK soundness game. Therefore,  $\Pr[\mathcal{C} \text{ wins}] \geq \Pr[\mathcal{C}_{\mathcal{A},2}^{\text{BaP}}]$ . However, since the  $zkSNARK$  is sound, we have that  $\Pr[\mathcal{C}_{\mathcal{A},2}^{\text{BaP}}] \leq \Pr[\mathcal{C} \text{ wins}] \leq \text{Adv}_{\text{POK}, \mathcal{C}}^{\text{sound}}(\kappa)$ .

The proof of the theorem follows since

$$\text{Adv}_{\mathcal{A}}^{\text{BaP}}(\kappa) \leq \Pr[\mathcal{C}_{\mathcal{A},1}^{\text{BaP}}] + \Pr[\mathcal{C}_{\mathcal{A},2}^{\text{BaP}}].$$

■

## 3) USER BALANCE

To win the user balance game, an adversarial provider must recover a valid signing ephemeral key  $sk_{\mathcal{U}}^e$  and use it to sign a bitcoin transaction to itself. However, this is prevented by the unforgeability of the signature scheme. Neither can the

provider post a transaction containing a smaller reward as the user would abort and not post the transaction that releases the encryption key  $K$ . Finally, the one-wayness of the hash function and the zero-knowledge property of the  $zkSNARK$  guarantee that a malicious provider cannot obtain  $K$  from the received  $H(K)$  or the proof  $\pi$ . This intuition is captured by the following theorem.

*Theorem 4 (User Balance):* If Theorem 1 is true, the signature scheme is secure, the hash function is one-way and the  $zkSNARK$  is zero-knowledge, our rewarding scheme satisfies Definition 5.

*Proof:* The proof is similar to that of Theorem 3 and is omitted. ■

## 4) PRIVACY

If there was an adversary  $\mathcal{A}$  that could distinguish the ephemeral keys of the two users  $\mathcal{U}_1$  and  $\mathcal{U}_2$  this would imply an adversary that could break the blindness property of the underlying signature scheme. The second adversary would simply use  $\mathcal{A}$ 's guesses for the ephemeral keys of  $\mathcal{U}_1$  and  $\mathcal{U}_2$  to answer the challenge of the corresponding blind signature game. This is formalized below.

*Theorem 5 (Privacy):* If the blind signature scheme is secure, our rewarding scheme is user-private as per the Definition 6.

*Proof:* Let  $\mathcal{A}$  be an adversary that distinguishes between  $\mathcal{U}_1$  and  $\mathcal{U}_2$  in experiment  $\text{Exp}_{\mathcal{A}}^{\text{Priv}}$  with probability better than random guessing. If this is possible, an adversary  $\mathcal{B}$  exists that attacks the security of the blind signature scheme used during registration.

$\mathcal{B}$  simulates  $\mathcal{A}$  queries and waits until  $\mathcal{A}$  produces the ephemeral keys  $pk_{\mathcal{U}_i}^e$  for  $\mathcal{U}_i$ ,  $i = 0, 1$ . Then  $\mathcal{B}$  computes the corresponding hash values  $h_i = H(pk_{\mathcal{U}_i}^e)$  and submits them to its challenger in the signature indistinguishability game. Let  $\text{Sig}_{\mathcal{P}}(h_b)$  and  $\text{Sig}_{\mathcal{P}}(h_{1-b})$  be the signatures created by the signature oracle and returned to  $\mathcal{B}$ , for some random bit  $b$ .  $\mathcal{B}$  forwards these to  $\mathcal{A}$  to continue its game in the privacy experiment. When  $\mathcal{A}$  outputs its guess  $b'$ ,  $\mathcal{B}$  outputs the same guess. If  $\mathcal{A}$  guessed correctly (i.e.  $b' = b$ ) then  $\mathcal{B}$  can tell between the two blind signatures. Thus the advantage of  $\mathcal{A}$  in the privacy experiment is bounded by the advantage of an adversary in breaking the security of the blind signature scheme. ■

## VII. IMPLEMENTATION

In this section, we look at various implementation details of the protocol. We consider both a Bitcoin-based approach as well as smart contracts-based one.

### A. $zkSNARKs$ GENERATION

We start by reporting the time and memory required to run the algorithms of the  $zkSNARK$  proof system (recall Sections IV and V-E). The proof  $\pi$  attests to the correctness of the encryption key  $K$ , the validity of the application of the rewarding function  $\mathcal{R}$  on the data as well as the application of the user's signing key in the construction of the redeeming tag.

The xJsnark [36] framework was used to write our zkSNARK verification program. The resulting arithmetic circuit was then ported into libsnark [37]. We set the rewarding function on data  $m$  equal to  $R(m) = \max(u_{min}, \min(a_1 m + a_0, u_{max}))$ , for some constants  $u_{min}, u_{max}, a_0, a_1$ . Although the reward can be a fixed amount for user submissions, a more complicated expression was used to stress test the zkSNARK generator. The user data  $m$  was set to 1KB as if responding to a task that asked for 5 min temperature readings for a duration of more than a day.

Our findings show that the time to generate the proof  $\pi$  is 25.3 seconds on the user side, however the time to verify it is only 4.8ms. This is important from the provider’s point of view as the work per user is negligible. Thus, a provider can handle a large number of users with minimal overhead.

**B. COMMUNICATION OVERHEAD**

The fixed size of the proof (288 bytes) ensures that the communication overhead is dominated by the encrypted data  $C$  which had to be sent anyway (in either encrypted or plain form). This is analyzed in more detail below. Consider the message send from  $\mathcal{U}$  to  $\mathcal{P}$  (recall Figure 2):

$$\langle Enc_{\mathcal{P}}(C), h_K, val, \tau, \pi, pk_{\mathcal{U}}^e, Sig_{\mathcal{P}}(pk_{\mathcal{U}}^e), \sigma_e \rangle.$$

Both  $h_K$  and  $\tau$  are hash outputs so they can be taken to be equal to 160 bits or 20 bytes each. The value  $val$  of the data can fit in a 64-bit word, so this contributes another 8 bytes to the total. Bitcoin is based on the use of elliptic curve cryptography (secp256k1 curve) and in particular the ECDSA algorithm for signing. Thus the size of the public key  $pk_{\mathcal{U}}^e$  is 33 bytes, while the size of the resulting signature  $\sigma_e$  is bounded by 73 bytes. Using RSA as our blind signature scheme, the signature  $Sig_{\mathcal{P}}(pk_{\mathcal{U}}^e)$  of the provider contributes another 128 bytes (however elliptic curve variants or less expensive blind schemes can be used instead). Thus, ignoring the size of the encrypted data, the user must send 570 bytes, which also includes the size of the zkSNARK proof  $\pi$ .

The remaining overhead comes from  $Enc_{\mathcal{P}}(C)$ . However, instead of encrypting  $C$  with the provider’s public key one can encrypt a symmetric key and use this to encrypt the remaining data. Thus, public key encryption can be reduced to encrypting just a single key instead of the remaining data. As the data has to be sent anyway, the encryption overhead is minimal.

**C. BITCOIN IMPLEMENTATION**

We now turn our attention to the actual bitcoin transactions. The protocol transactions ( $T_{\mathcal{U} \rightarrow \mathcal{P}}$  and  $T_{\mathcal{P} \rightarrow \mathcal{U}}$ ) were built and tested on Bitcoin Core version 0.21.1 using the Segregated Witness (SegWit) transaction format. In particular, we used the bech32 (native SegWit) address type in order to minimize the size of the transactions posted in the blockchain.

The size of the complete transaction is 300 bytes and the size of the witness data is 220 bytes which includes the size of the key and the signature (for more details see [38]). This allows us to compute the virtual size of the transaction which

is given by the expression  $vsize = \lceil (btxs * 3 + ttxs) / 4 \rceil$ , where  $btxs$  is the base transaction size (which does not include the witness data), and  $ttxs$  is the complete transaction size. Hence the virtual size of each transaction is small and amounts to just 135 bytes.

*Cost estimation:* Adding a transaction to the bitcoin blockchain requires a transaction fee which is calculated in Satoshis/byte. The fee depends on the size of the transaction but also the time it takes to be mined. For example, a delay  $d$  means that it takes  $d$  blocks to mine the transaction. Hence bigger transaction delays lead to less Satoshis payed per byte and hence smaller overall cost. Table 3 demonstrates the cost of the Bitcoin implementation based on the Satoshis/byte values according to [39]. Funding and redeeming correspond to transactions  $T_{\mathcal{P} \rightarrow \mathcal{U}}$  and  $T_{\mathcal{U} \rightarrow \mathcal{P}}$ , respectively (Section V-F).

**TABLE 3. Cost of Bitcoin transactions based for a delay of  $d$  blocks.**

	$d = 1$	$d = 3$	$d = 5$
Fund	0.3501 USD	0.1751 USD	0.1167 USD
Redeem	0.396 USD	0.198 USD	0.132 USD

**D. SMART CONTRACTS IMPLEMENTATION**

The smart contract was implemented in solidity 0.7.6 [40] and tested on the Ethereum Ropsten test network. The smart contract consists of two main functions.

- 1) *Deploy & Fund:* The smart contract is first deployed and then funded by the provider  $\mathcal{P}$  with a deposit equal to the value of the data. This basically expresses  $\mathcal{P}$ ’s willingness to pay for the key received by the user  $\mathcal{U}$ . The function changes the smart contract’s state and allows  $\mathcal{U}$  to reveal the key.
- 2) *Redeem:* This is called by  $\mathcal{U}$  in order to reveal the key to the smart contract. If the key matches the hash value stored in the contract, the deposit is transferred to the user.

*Cost estimation:* Table 4 shows the gas cost of deploying the smart contract and executing the main functions for a key  $K$  of size 128 bits. As it can be seen, the cost of deploying the smart contract is the most expensive one.

**TABLE 4. Smart contract Transaction Gas.**

Transaction	Deploy & Fund	Redeem
Gas	481345	55245

Similarly to Bitcoin blockchain, fees determine the time it takes for the transaction to be added to the Ethereum blockchain. In this case, however, fees of Ethereum transactions are calculated in Gwei/gas; more Gwei per gas means a faster processing time. Table 5 demonstrates the transaction costs based on the Gwei/gas values according to [41].

**E. COMPARISON**

Comparing Tables 3 and 5, we see that the Ethereum implementation is at least two orders of magnitude more

**TABLE 5. Cost of smart contract based on the time transactions are added to the Ethereum blockchain.**

	5 Minutes	15 Minutes	20 Minutes
Deploy & Fund	31.74 USD	30.61 USD	29.47 USD
Redeem	3.6427 USD	3.5126 USD	3.3825 USD

expensive than the corresponding Bitcoin implementation. This is attributed to the simpler Bitcoin *Pay-to-Script-Hash* transactions used to pay for the release of the secret key. On the other hand, even a simple smart contract as the one used in this work is extremely costly. This also suggests the superiority of Bitcoin implementation compared to [11], [12], [13], and [14] that use the smart contract to exchange data as well. Hence the cost of these solutions depends on the size of the data transmitted and does not remain fixed as in our case.

Finally, Table 1 shows a comparison of our protocols with other existing crowd-sensing protocols regarding the confidentiality and strong fairness guarantees. This analysis (in addition to the cost advantage) shows the practicality of our approach in the scenarios envisioned by crowd-sensing applications.

## VIII. CONCLUSION

In this work, we developed a privacy-preserving rewarding framework that can be used to incentivize and increase user participation in crowd-sensing applications. With the help of our framework users can submit data collected with their smart devices and obtain *rewards* in the form of bitcoin payments. Our protocol guarantees the anonymity of submissions without sacrificing accountability. Indeed one of the key requirements in our work is to prevent *double-redeeming* attacks in which a user may attempt to obtain multiple rewards for the same data. Our proposal prevents this malicious behavior without giving up anonymity of transactions. Thus user submissions cannot be distinguished and rewards remain unlinkable. More importantly, our protocol guarantees the *fairness* of the exchange as neither the user nor the provider can cheat each other. Finally, our protocol is highly efficient as most of the steps take place *offchain* and only the actual payment exchange uses the blockchain network. We have implemented our protocol in both Bitcoin and Ethereum blockchains. Since we only rely on simple *Pay-to-Script-Hash* transactions, as opposed to complex smart contracts used in prior works, the viability of our approach is ensured.

## ACKNOWLEDGMENT

The authors would like to thank the reviewers for their comments that helped improve the paper considerably, and also would like to thank Karim Elmaghraby for his implementation of the Bitcoin and Ethereum scripts.

## REFERENCES

[1] G. Chatzimilioudis, A. Konstantinidis, C. Laoudias, and D. Zeinalipour-Yazti, "Crowdsourcing with smartphones," *IEEE Internet Comput.*, vol. 16, no. 5, pp. 36–44, Sep. 2012.

[2] D. Christin, "Privacy in mobile participatory sensing: Current trends and future challenges," *J. Syst. Softw.*, vol. 116, pp. 57–68, 2016.

[3] I. Krontiris, C. F. Freiling, and T. Dimitriou, "Location privacy in urban sensing networks: Research challenges and directions," *IEEE Wireless Commun.*, vol. 17, no. 5, pp. 30–35, Oct. 2010.

[4] F. Restuccia, S. K. Das, and J. Payton, "Incentive mechanisms for participatory sensing: Survey and research challenges," *ACM Trans. Sensor Netw.*, vol. 12, no. 2, pp. 1–40, May 2016.

[5] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 4, pp. 593–610, Apr. 2000.

[6] P. H. Drielsma and S. Mödersheim, "The ASW protocol revisited: A unified view," *Electron. Notes Theor. Comput. Sci.*, vol. 125, no. 1, pp. 145–161, Mar. 2005.

[7] T. Dimitriou, "Fair and private bitcoin rewards: Incentivizing participation in crowd-sensing applications," in *Proc. IEEE Int. Conf. Decentralized Appl. Infrastruct. (DAPPS)*, Aug. 2020, pp. 120–125.

[8] C. Tanas, S. Delgado-Segura, and J. Herrera-Joancomart, "An integrated reward and reputation mechanism for MCS preserving users privacy," in *Proc. Int. Workshop Data Privacy Manage. Secur. Assurance*, 2015, pp. 83–99.

[9] J. Chen and Y. Xue, "Bootstrapping a blockchain based ecosystem for big data exchange," in *Proc. IEEE Int. Congr. Big Data (BigData Congr.)*, Jun. 2017, pp. 460–463.

[10] W. Feng and Z. Yan, "MCS-Chain: Decentralized and trustworthy mobile crowdsourcing based on blockchain," *Future Gener. Comput. Syst.*, vol. 95, pp. 649–666, Jan. 2019.

[11] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J.-N. Liu, Y. Xiang, and R. H. Deng, "CrowdBC: A blockchain-based decentralized framework for crowdsourcing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 6, pp. 1251–1266, Jun. 2019.

[12] Y. Lu, Q. Tang, and G. Wang, "ZebraLancer: Private and anonymous crowdsourcing system atop open blockchain," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 853–865.

[13] H. Duan, Y. Zheng, Y. Du, A. Zhou, C. Wang, and M. H. Au, "Aggregating crowd wisdom via blockchain: A private, correct, and robust realization," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2019, pp. 1–10.

[14] W. Dai, C. Dai, K.-K.-R. Choo, C. Cui, D. Zou, and H. Jin, "SDTE: A secure blockchain-based data trading ecosystem," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 725–737, 2020.

[15] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiaainen, S. Capkun, and A.-R. Sadeghi, "Software grand exposure: SGX cache attacks are practical," in *Proc. 11th USENIX Workshop Offensive Technol.*, 2017, pp. 1–12.

[16] S. Dziembowski, L. Ecekey, and S. Faust, "FairSwap: How to fairly exchange digital goods," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 967–984.

[17] J. Zhang, W. Cui, J. Ma, and C. Yang, "Blockchain-based secure and fair crowdsourcing scheme," *Int. J. Distrib. Sensor Netw.*, vol. 15, no. 7, Jul. 2019, Art. no. 155014771986489.

[18] J. Wang, M. Li, Y. He, H. Li, K. Xiao, and C. Wang, "A blockchain based privacy-preserving incentive mechanism in crowdsensing applications," *IEEE Access*, vol. 6, p. 17545–17556, 2018.

[19] S. Delgado-Segura, C. Pérez-Solá, G. Navarro-Arribas, and J. Herrera-Joancomartí, "A fair protocol for data trading based on bitcoin transactions," *Future Gener. Comput. Syst.*, vol. 107, pp. 832–840, Jun. 2020.

[20] I. Bentov and R. Kumaresan, "How to use bitcoin to design fair protocols," in *Advances in Cryptology—CRYPTO 2014 (Lecture Notes in Computer Science)*, vol. 8617. Berlin, Germany: Springer, 2014.

[21] E. Heilman, L. AlShenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "TumbleBit: An untrusted bitcoin-compatible anonymous payment hub," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2017, pp. 1–36.

[22] M. Campanelli, R. Gennaro, S. Goldfeder, and L. Nizzardo, "Zero-knowledge contingent payments revisited: Attacks and payments for services," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 229–243.

[23] G. Maxwell. (2015). *Zero Knowledge Contingent Payment*. Accessed: Mar. 24, 2022. [Online]. Available: [https://en.bitcoin.it/wiki/Zero\\_Knowledge\\_Contingent\\_Payment](https://en.bitcoin.it/wiki/Zero_Knowledge_Contingent_Payment)

[24] D. Tassos, "Privacy-respecting reward generation and accumulation for participatory sensing applications," *Pervasive Mobile Comput.*, vol. 49, pp. 139–152, Sep. 2018.

- [25] T. Dimitriou, T. Giannetsos, and L. Chen, "REWARDS: Privacy-preserving rewarding and incentive schemes for the smart electricity grid and other loyalty systems," *Comput. Commun.*, vol. 137, pp. 1–14, Mar. 2019.
- [26] S. Saroiu and A. Wolman, "I am a sensor, and I approve this message," in *Proc. 11th Workshop Mobile Comput. Syst. Appl.*, 2010, pp. 37–42.
- [27] U. Gadiraju, R. Kawase, S. Dietze, and G. Demartini, "Understanding malicious behavior in crowdsourcing platforms: The case of online surveys," in *Proc. 33rd Annu. ACM Conf. Hum. Factors Comput. Syst.*, Apr. 2015, pp. 1631–1640.
- [28] T. Dimitriou, "Decentralized reputation," in *Proc. 11th ACM Conf. Data Appl. Secur. Privacy*, Apr. 2021, pp. 119–130.
- [29] F. Restuccia, N. Ghosh, S. Bhattacharjee, S. K. Das, and T. Melodia, "Quality of information in mobile crowdsensing: Survey and research challenges," *ACM Trans. Sensor Netw.*, vol. 13, no. 4, pp. 1–43, Nov. 2017.
- [30] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: Characterizing payments among men with no names," in *Proc. Conf. Internet Meas. Conf.*, Oct. 2013, pp. 127–140.
- [31] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 1–16.
- [32] M. Bellare, G. Fuchsbauer, and A. Scafuro, "NIZKs with an untrusted CRS: Security in the face of parameter subversion," in *Advances in Cryptology—ASIACRYPT 2016*. Berlin, Germany: Springer, 2016, pp. 777–804.
- [33] *Pay-to-Script Hash*. Accessed: Mar. 24, 2022. [Online]. Available: [https://en.bitcoinwiki.org/wiki/Pay-to-Script\\_Hash](https://en.bitcoinwiki.org/wiki/Pay-to-Script_Hash)
- [34] Gavin Wood. (2020). *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Accessed: Mar. 24, 2022. [Online]. Available: <https://ethereum.github.io/yellowpaper/paper.pdf>
- [35] *Transaction Confirmation*. Accessed: Dec. 3, 2022. [Online]. Available: [https://en.bitcoinwiki.org/wiki/Transaction\\_confirmation](https://en.bitcoinwiki.org/wiki/Transaction_confirmation)
- [36] A. Kosba, C. Papamanthou, and E. Shi, "xJsnark: A framework for efficient verifiable computation," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 944–961.
- [37] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von Neumann architecture," in *Proc. 23rd USENIX Conf. Secur. Symp.*, 2014, pp. 781–796.
- [38] E. Lombrozo, J. Lau, and P. Wuille. *BIP141: Segregated Witness (Consensus Layer)*. Accessed: Mar. 24, 2022. [Online]. Available: <https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki>
- [39] *Bitcoin Satoshi—Fee Estimates*. Accessed: Mar. 29, 2022. [Online]. Available: <https://statoshi.info/dashboard/db/fee-estimates>
- [40] *Solidity 0.7.6*. Accessed: Mar. 31, 2022. [Online]. Available: <https://docs.soliditylang.org/en/v0.7.6/>
- [41] *Ethereum Gas—Fee Estimates*. Accessed: Mar. 31, 2022. [Online]. Available: <https://ethgasstation.info/>



**TASSOS DIMITRIOU** (Senior Member, IEEE) is currently a Professor at the Department of Computer Engineering, Kuwait University. Prior to that, he was an Associate Professor at Athens Information Technology, Greece, where he was leading the Algorithms and Security Group, and an Adjunct Professor at Carnegie Mellon University, USA, and Aalborg University, Denmark. He conducts research in areas spanning from the theoretical foundations of cryptography to the design and implementation of efficient and secure communication protocols with emphasis on authentication and privacy for various types of networks (ad-hoc, sensor and ubiquitous networks, RFID, and smart grid), security architectures for wireless networks, and the development of secure applications for networking and electronic commerce. His research in the above fields has resulted in numerous publications, some of which received distinction, and numerous invitations for talks in prestigious conferences. He is a Senior Member of ACM and a Fulbright Fellow. He has been a Distinguished Lecturer of ACM. For more information visit the link (<http://tassosdimitriou.com/>).



**ANTONIS MICHALAS** received the Ph.D. degree from Aalborg University, Denmark. He currently works as an Associate Professor at the Department of Computing Sciences, Tampere University, Finland, where he also co-leads the Network and Information Security Group (NISEC). The group comprises Ph.D. students, professors, and researchers. Group members mainly conduct research in the areas of applied cryptography, provable security, and privacy. Apart from his research work at NISEC, as an Associate Professor, he is actively involved in the university's teaching activities while his role also expands to student supervision and research projects coordination. Furthermore, he has published a significant number of papers in field-related journals and conferences.

...