Privacy-respecting Reward Generation and Accumulation for Participatory Sensing Applications

Tassos Dimitriou, *Senior Member, IEEE*

Abstract

Participatory or crowd-sensing applications process sensory data contributed by users and transform them to simple visualizations (such as for example noise or pollution levels) that help create an accurate representation of the surrounding environment. Although contributed data is of great interest to individuals, the involvement of citizens and community groups, however, is still limited. Hence, incentivizing users to increase participation seems crucial for the success of participatory sensing.

In this paper, we develop a privacy-preserving rewarding scheme which allows campaign administrators to reward users for the data they contribute. Our system of anonymous tokens allow users to enjoy the benefits of participation while at the same time ensuring their anonymity. Moreover, rewards can be accumulated together thus further increasing the level of privacy offered by the system. Our proposal is coupled with a security analysis showing the privacy-preserving character of the system along with an efficiency analysis demonstrating the feasibility of our approach in realistic deployment settings.

Index Terms

Rewards, Incentives, User privacy, Anonymity, Credential systems, Zero-knowledge proofs, Participatory sensing, Crowd sensing.

I. Introduction

Participatory sensing (or crowd-sensing) is a new paradigm for data collection and knowledge representation that has been enabled by the proliferation of mobile devices with sensing capabilities. Using the sensors embedded in smart phones and other similar devices, users can interact with the environment and create a better understanding of people's activities and their surroundings. Participatory sensing applications are numerous; participants can monitor and report environmental conditions such as noise or air pollution, they can document health related issues or physical activities, they can monitor road and traffic conditions, and so on [1].

The key factor for the success of crowd-sensing applications is user participation. Even though the use of smart phones and tablets has made participation easy, voluntary commitment of participants is still limited. This is mostly due to the fact that users do not see any immediate benefits for contributing the sensing capabilities of their mobile devices. Thus incentivizing users to increase the amount and the quality of the data they submit seems necessary for the success of participatory sensing applications.

An important research direction in this area has been to come up with the right incentives that would make people more involved in the data collection process. This is typically done by defining appropriate mechanisms which can be used to reward users for their sensed data while at the same time maximizing the utility of the collected data. Such mechanisms can be specific to a particular application [2], in which case they do not generalize easily, or they can be agnostic to the application as long as the data collection process satisfies the assumptions of the mechanism. In this last case, rewards are typically based on game theoretical approaches [3], [4] whose aim is to increase participation of users making rational choices for their own benefit.

In this work, however, we will not focus on the incentive mechanisms that can be used to solicit user participation; as mentioned above this is a problem closer to the theory of microeconomics, where demand and supply decide the value of the provided data. Instead, we will focus on the equally important problem of coming up with appropriate mechanisms to *reward* users for the sensed data they submit. This problem has been largely overlooked as most works focus either on the privacy of reported data or the incentives to increase user participation but not both [5], [6], [7]. Rewarding is a challenging problem as only authenticated users should benefit from the use of such services. On the other hand, the identity of the user or other contextual information should not serve as an identifier that can be used to filter the user's transactions or link it to the rewards collected.

Contributions: In this work, we propose a privacy-preserving scheme of anonymous tokens which can be used by the service provider as a means to reward those users for the data they contributed. During data submission, users earn credits by means of anonymous tokens. Contrary to prior works, however, rewards for different data submissions can be *accumulated* together. This increases the level of privacy as different rewards can be aggregated to a single token, thus reducing the probability of tracking users by the tokens they carry or spend.

Our proposal can be thought as complementary to all approaches that try to enhance user privacy in participatory sensing applications (for a survey see [7]). As most applications collect spatio-temporal information to annotate sensor data, user

For up-to-date contact information please visit http://tassosdimitriou.com. Email: tassos.dimitriou@ieee.org.

privacy is put at risk. Hence protection mechanisms are required to ensure that participants' identities cannot be revealed when downloading sensing tasks or reporting sensed data back to the campaign administrators. Our solution integrates nicely with all such proposals whose goal is to ensure anonymity during data submission; every time user data is submitted, our protocol can make sure that an appropriate reward will be credited to an unlinkable token which can be used to aggregate further rewards given to the same (anonymous) user.

We thoroughly analyze the security of our proposal and we show that such tokens are indeed privacy preserving. In particular, token accumulation does not leak any information about the identity of users and cannot be used in any way by the provider to profile them. Thus our protocols ensure the privacy and unlinkability of transactions. Our work shows that more advanced rewarding mechanisms can be integrated in participatory sensing frameworks, while at the same time offering strong privacy guarantees during the rewarding phase.

Finally, we analyze the efficiency aspects of our proposal; our findings show that our protocols do not incur any significant overhead, and thus can be easily handled by modern-day user devices such as smartphones.

Organization: The remainder of the paper is structured as follows. In Section II, we review related work on rewarding schemes in urban sensing applications. In Section III, we introduce our privacy model, we discuss system requirements and assumptions, and we review the basic tools we will be using throughout this work. The details of our privacy-respecting rewarding scheme can be found in Section IV. Its security and efficiency aspects are analyzed in Section V. Section VI discusses possible extensions, while Section VII concludes the paper.

II. Related work

There are many works in the literature whose goal is to motivate users to participate in sensing campaigns. Understanding the reasons why some people may respond to monetary rewards while others may be willing to participate for free, provided they have access to sensed data, is an interesting research question in the area (for a recent survey see [8]). However, there is little work that focus on the actual rewarding mechanism that can be used to reward users for the data they provide. Our work aims to fill this gap.

One of the mechanisms that have been used to increase high-quality contributions from users (but with no emphasis on privacy) is the use of micro-payments. This concept has been studied by Reddy *et al.* [9] which have shown that micro-payments is an effective tool for increasing the quantity of submissions as well as the quality of sensed data.

In [10], Li and Cao proposed two privacy-aware incentive schemes for participatory sensing applications. The first scheme relies on a Trusted Third Party to ensure user privacy. The second is a TTP-free scheme which uses blind signatures, partially blind signatures and commitment techniques to construct tokens (request, report and credit tokens) that can be used in various phases of the protocol. As the authors mention, "for each credit token, the node waits a random time and then, using its real identity, deposits the token to the collector. The collector maintains a credit account for each node in the system, and it updates the depositing nodes credit account accordingly." This severely affects privacy as a reward can be linked to a given task/submission. Thus, the level of security offered is questionable. This problem is solved in our proposal by (i) having the user (and not the provider) maintaining the sum of rewards collected thus far (through aggregation), and (ii) allowing the user to redeem any portion of the amount kept in the user's token.

Reward tokens have also been used by Dimitriou and Krontiris [11] in reverse auctions as a way to incentivize users in participatory sensing applications to submit better quality data. The reverse auction takes place among users (sellers) and data requester (buyers) of sensing data. This mechanism is more attractive than regular auctions as it eliminates the need for the requester to set or guess the price which users consider reasonable for their data. The protocol is very efficient in practice, it guarantees bidders' privacy and unlinkability of tokens and provided data, however the tokens cannot be aggregated together.

Rewarding schemes can also be used to maintain the reputation of participating users. This is the underlying idea in IncogniSense [12] where contributed data are rated by associating a reputation score to each user. The system utilizes periodic pseudonyms generated using RSA blind signatures and relies on reputation transfer between these pseudonyms. The reputation transfer process has an inherent trade-off between anonymity protection and loss in reputation. The main problem lies in the fact that reputation tokens earned with the current pseudonym must be sent back to the provider in order to be credited to the next pseudonym. Thus lack of aggregation by the user opens up the possibility for the same inference attacks as in the case of [10].

Changing the domain of applicability, Rupp *et al.* [13] proposed a token-based mechanism to be used in public transportations systems which also enables aggregation of refunds on suitably defined tokens. The scheme is very efficient in practice however it based on the assumption that the set of diferent refund amounts is limited since these are basically encoded in the public key of the provider. This is not required in our proposal which can support arbitrary amounts.

In [14], the authors propose a privacy-preserving loyalty system which allows vendors to build customer profiles from tokens submitted by the same user. The emphasis here is on the ability of the vendor to link together tokens, once permitted by the user. The scheme is based on partially-blind signatures which can be used to embed information that will enable the vendor to perform aggregate verification of signed messages bearing the same agreed public information.

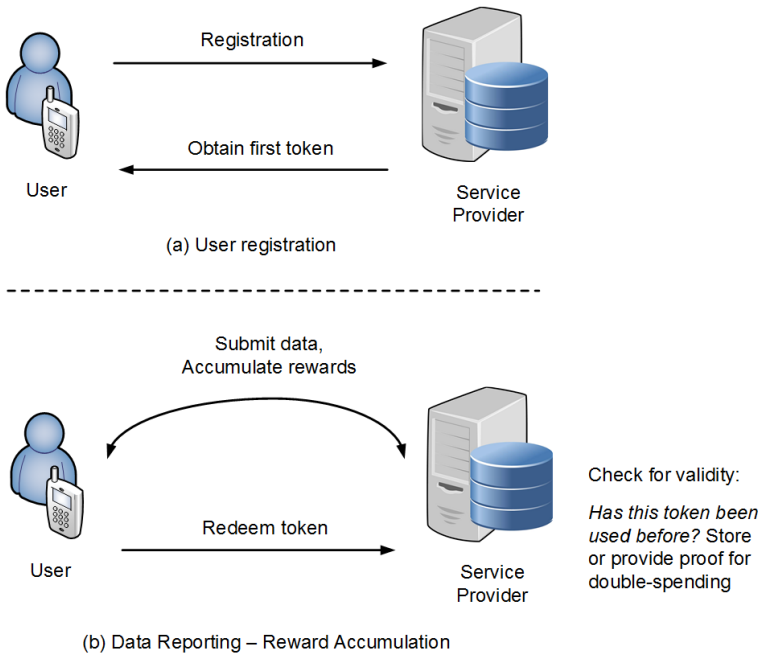

Fig. 1. Acquisition of sensing data, bidding and reporting.

In [15], Yang *et al.* developed a privacy-preserving communication and reward architecture for Vehicle-to-Grid networks. The system is based on ID restrictive partially blind signatures to create reward tokens that can be redeemed at different times. Lack of accumulation, however, suggests that a vehicle can be profiled based on the rewards it received, since redemption of multiple rewards at the same time may leak information about the user's whereabouts.

III. Security Model and Main Tools

Our main goal is to develop a scheme that will allow users to collect rewards for the data they provide. The main entities in our system are the users $\mathcal{U}$ and the provider $\mathcal{P}$ who rewards users for their data. As it can be seen in Figure 1, there is an initial phase where users register and obtain their first token (or reward accumulator) of initial value zero. At any time, the users have to maintain *only one* token. Hence in the main operational phase, users earn rewards for data they submit and accumulate them in their reward token. As mentioned in the introduction, our proposal integrates easily with all solutions whose goal is to ensure user anonymity during data submission. Hence, our focus here will be only on the rewarding mechanism. It should be clear that a careful design of this mechanism is required in order to ensure the privacy and anonymity of users who are accumulating the rewards, namely:

- Rewards can be collected or redeemed only by legitimate users. Any attempt to double-spend a token or accumulate the same reward more than once should be easily detectable. Additionally, users should not be able to forge their own rewards or construct new tokens.
- Reward accumulation should not leak any information about the underlying user, and rewards should be unlinkable to each other (or the data reported by users).

In what follows, we list the algorithms and protocols that enable users to register into the system, obtain tokens and accumulate rewards.

- $\mathsf{KGen}(1^k)$ is a probabilistic algorithm that on input a security parameter k, it generates the provider's public and private keys (pk_P, sk_P) along with other system parameters.
- $\mathsf{Register}(\mathcal{U}, \mathcal{P})$ is protocol executed between a user $\mathcal{U}$ and $\mathcal{P}$. The outcome of this protocol is a secret identity ID_u and public identity $I = g^{ID_u}$ that is associated with the user and stored in a provider's database. ID_u will be used as evidence when a user is caught double-spending or trying to accumulate an already accounted reward.
- $\mathsf{GetToken}(\mathcal{U}(ID_u), \mathcal{P})$ is a protocol executed between a user with secret id ID_u and $\mathcal{P}$. Upon successful execution, the user obtains an initial reward token τ of value zero with the user's ID embedded in it and bearing the provider's signature.

Experiment $\mathbf{Exp}^{Priv}_{\mathcal{A},\mathcal{U}}$:

Setup phase: Adversary $\mathcal{A}$ creates and publishes the system parameter param. A set $\mathcal{U}$ of users is initialized by means of calls to the Register and GetToken protocols in order to participate in the reward collection process with the adversarial provider $\mathcal{A}$.

Learning phase: Adversary asks any number of users to accumulate rewards by means of calls to AccumulateToken and any amount v. The adversary may ask users to do this multiple times.

Challenge phase:

- $\mathcal{A}$ selects two users $\mathcal{U}_0$ and $\mathcal{U}_1$. It is then given access to a user oracle.
- $b \xleftarrow{R} \{0,1\}$. Adversary $\mathcal{A}$ interacts with user $\mathcal{U}_b$ by means of the AccumulateToken oracle.

Post-Challenge phase: $\mathcal{A}$ may ask both $\mathcal{U}_0$ and $\mathcal{U}_1$ to further accumulate rewards multiple times by means of calls to AccumulateToken.

$\mathcal{A}$ outputs a guess bit b'. Adversary is successful if $b = b'$.

Fig. 2. Privacy experiment.

- AccumulateToken$(\mathcal{U}(\tau), v, \mathcal{P})$ is a protocol executed between a user possessing a signed reward token τ and $\mathcal{P}$. Upon successful execution, the value of the token is increased by the amount v.
- RedeemToken$(\mathcal{U}(\tau), \mathcal{P})$ is a protocol executed between a user possessing a signed reward token τ and $\mathcal{P}$. Upon successful execution, the token is fully exchanged with services and products offered by the provider of value equal to the value stored in the token. This function can be easily extended to one supporting *partial* reward redemption as explained in Section VI.
- CheckDoubleSpending$(\mathcal{U}(\tau), \mathcal{P})$ is a protocol executed by P to test if a token is invalid. If this is the case, the secret identity ID_u of the user is revealed. This function is executed during accumulation or redemption of a token.

A. Modeling user privacy

In this section, we discuss the capabilities of an adversarial provider $\mathcal{A}$ whose goal is to identify a user when trying to accumulate rewards. Essentially, with the exception of the registration phase, updating token values should not leak any information about the user. Thus, rewards should be unlinkable to each other and to any data reported by users. Otherwise, users risk leaking additional details about spending patterns, types of services they request, etc.

To model this, we will consider an indistinguishability experiment in which two users $\mathcal{U}_0$ and $\mathcal{U}_1$ interact with an adversarial provider $\mathcal{A}$. Initially, the adversary asks the two users to register and then accumulate any number of rewards to their respective tokens. Once this learning phase is over, a user $\mathcal{U}_b$ is selected according to a random bit b unknown to the adversary and asked to use its token to accumulate a predefined amount v. The scheme will be privacy-preserving if the adversary is not able to identify the bit b with probability better than random guessing. Intuitively, the adversary should not be able to link users to the rewards they received under the condition that all other actions (registration, get first token, etc.) are controlled by $\mathcal{A}$.

This is formalized by the experiment shown in Figure 2. The adversary interacts with users by means of oracles Register, GetToken which couple the main operations allowed in the system. Typically, the interaction between the adversary and the users will occur only by using such oracles which will help us modeling the capabilities of the adversary in a real attack.

Definition 1: A reward accumulation system is privacy-preserving if an adversary $\mathcal{A}$ cannot predict with more than $1/2 + \epsilon$ probability the bit b in the game described below, where ϵ is a negligible quantity.

An implicit assumption here is that the adversary does not have access to the internal state of users since otherwise it could act on behalf of users and easily win the above game. For example, the adversary could see the information retained by users, re-randomizations performed, etc. before and after the challenge phase and thus infer the user chosen by the oracle.

B. Side channels undermining user privacy

In this work, we consider an adversarial provider who follows the protocol but tries to infer information about the users by monitoring data exchanged in the various protocol runs. Additionally, we will be ignoring background information and statistics that can be used to infer information about users. If it is known, for example, that some users are more willing to use the system than others then certainly the anonymity set will be small. For the same reason, we will be assuming that users interact with the provider through some form of an *anonymity* network, since if the IP address of a user is visible when the user obtains a reward, then it can be easily linked to the user identity submitted during the registration phase. Hence additional

mechanisms are required to ensure that a network connection remains anonymous (this is a standard assumption in all works where users directly interact with an adversarial provider). One such mechanism is the anonymity network TOR [16].

In summary, our protocol ensures unlinkability and untraceability of rewards provided such side channels are eliminated. Using only information available in the protocol runs, we will show that our system is privacy preserving.

C. Main tools

In what follows we describe the main tools we will be using in our proposal. We assume the existence of a Setup algorithm which upon input a security parameter 1^k outputs the parameters for two groups $G = \langle g \rangle$ and $\mathbf{G} = \langle \mathbf{g} \rangle$ of prime order $q = O(2^k)$ that have an efficiently computable bilinear pairing e.

1) Pairings: A function $e : G \times G \to \mathbf{G}$ is called a bilinear pairing if the following hold:

- *Bilinearity*: For all $P, Q \in G$ and for all $a, b \in Z$, $e(P^a, Q^b) = e(P, Q)^{ab}$.
- *Non-degeneracy*: There exist $P, Q \in G$ such that $e(P, Q) \neq 1$, where 1 is the identity in $\mathbf{G}$.
- *Efficiency*: e can be efficiently computed.

2) Pedersen commitment scheme: A commitment scheme allows a user to commit to a value m in a way that reveals no information about m. In the commitment phase, the committer uses algorithm $\mathsf{Commit}(m, r)$, which takes as input a message m and an unpredictable random number r, to produce a commitment $c = \mathsf{Commit}(m, r)$. The commitment scheme is secure if it is both binding and hiding. The receiver should get no information about m before the opening phase, while the "binding" property ensures that a malicious committer cannot find values $m' \neq m$ and r' such that $\mathsf{Commit}(m, r) = \mathsf{Commit}(m', r')$.

The Pedersen commitment scheme [17] is an example of commitment scheme that can be generalized to commit to a number of messages $m_1, \ldots, m_n$. Let G be a group of prime order $q = O(2^k)$, with k being the security parameter, and $g, g_1, g_2, \ldots, g_n$ generators of G. To commit to $m_1, \ldots, m_n$, the user produces $\mathsf{Commit}(m_1, \ldots, m_n, r) = g^r \Pi_{i=1}^n g_i^{m_i} \mod p$.

3) Camenisch and Lysyanskaya signature scheme: Another tool that we will use in our proposal is the signature scheme proposed by Camenisch and Lysyanskaya in [18]. While this scheme can be used to construct anonymous credentials, we will build upon it to provide for privacy-preserving reward tokens. The signature scheme can be used to sign any number of message blocks $m_0, m_1, \ldots, m_n$ for which a commitment is known. In our scheme we will be using the case $n = 3$, but we describe the general case below.

Run the Setup algorithm to generate $(q, G, \mathrm{G}, g, \mathrm{g}, e)$. Let $sk = (x, y, \{z_i\})$ be the secret key of the signer and let $X = g^x$, $Y = g^y$, $Z_i = g^{z_i}$ and $W_i = Y^{z_i}$, for $i = 1, \ldots, n$. The public key pk consists of the values $(q, g, X, Y, \{Z_i\}, \{W_i\})$.

Suppose $C = g^{m_0} \Pi_{i=1}^n Z_i^{m_i}$ is a commitment to messages $m_0, m_1, \ldots, m_n$. To sign C, the user first has to prove knowledge of the opening of the commitment. Then, to compute the signature σ on C, the signer does the following:

- Pick random $\alpha \in Z_q$.
- Set $a = g^\alpha$ and $A_i = a^{z_i}$ for $i = 1, \ldots, n$.
- Set $b = a^y$ and $B_i = A_i^y$ for $i = 1, \ldots, n$.
- Set $c = a^x C^{\alpha x y}$.

The signature on committed message $(m_0, , \ldots, m_n)$ is given by $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$. To verify it, a user with possession of the public key pk has to check the following:

- $\{A_i\}$ were formed correctly, i.e. $e(a, Z_i) = e(g, A_i)$.
- b and $\{B_i\}$ were formed correctly, i.e. $e(a, Y) = e(g, b)$ and $e(A_i, Y) = e(g, B_i)$.
- c was formed correctly, i.e. $e(X, a) \cdot e(X, b)^{m_0} \cdot \Pi_{i=1}^n e(X, B_i)^{m_i} = e(g, c)$.

4) Zero-knowledge proofs of knowledge: A zero-knowledge (ZK) proof is an interactive protocol between a prover P and a verifier V, where the prover tries to convince the verifier about the validity of a statement without the verifier learning anything beyond this fact.

A ZK proof protocol must be complete, sound and zero-knowledge [19]. Soundness suggests the existence of a knowledge extractor algorithm which can extract a witness to the validity of the statement from every (potentially dishonest) prover P^* succeeding in the protocol with non-negligible probability. Zero-knowledge means that for every (potentially dishonest) verifier V^*, there exists a simulator which can generate a transcript that is indistinguishable from a real transcript of an interaction between V^* and P.

We will be following standard notation in describing such a proof. For example, we will denote by $\pi = PK\{(r, u, v) : C = g^r h^u \wedge I = g^v\}$ an interactive proof where the prover tries to convince the verifier that it knows r, u and v such that $C = g^r h^u$ and $I = g^v$. The variables r, u, v inside parentheses will denote private values, while C and I on the right will constitute public information available to the verifier.

IV. Rewards generation and accumulation

Here we develop our scheme that will allow users to collect rewards for the data they provide. What distinguishes our scheme from past works is that rewards can be accumulated to a single token. Hence users do not have to maintain a collection of different tokens for data submitted at different time periods. This makes rewarding easier and motivates users to participate in the data collection process. Another benefit of our approach is that the use of a single token makes tracking difficult; when the user tries to redeem the token, the provider cannot link it to a specific reward v for data submitted in the past as all these different rewards have all been aggregated together. Caution, however, needs to be taken in order to prevent double-spending attempts. For this, the ID of the user will be embedded in the token, allowing the provider to spot malicious users when they try to redeem (or accumulate) a reward twice.

The general structure of a reward token will be given by a tuple $\tau = (\langle rnd, tID, uID, val\rangle, \sigma)$, where rnd is a random quantity used to blind the other token values, tID is a token identifier that can be used to check for double-spending against the user ID uID, val is the value of the token accumulated thus far and σ is a signature on the values committed to the token. Effectively, tID will be linked to uID but masked by rnd to ensure that user tracking is not possible. Later on we will see how double-spending can also be prevented even though the token ID will change with every operation made on the token. The exact details are described in the following sections.

A. Key generation and Setup

The system is initialized with a call to $\mathsf{KeyGen}(1^k)$, where k is the security parameter. Once the group parameters $(q, G, \mathrm{G}, g, \mathrm{g}, e)$ are created (recall Section III-C3), the secret key of the provider becomes $sk = (x, y, \{z_i\})$, for random x, y and z_i, and its public key $pk = (q, g, X, Y, \{Z_i\})$, where $X = g^x$, $Y = g^y$, $Z_i = g^{z_i}$, for $i = 1, 2, 3$.

Without loss of generality we will assume that the values Z_i are also generators of Z_q. This is easy to achieve if q has the form $2p + 1$ for some other prime p. In that case, if g is a generator, then $Z_i = g^{z_i}$ is also a generator provided $gcd(z_i, \phi(q)) = 1$, where $\phi()$ is Euler's function. Thus, overall there are $\phi(\phi(q)) = p$ such generators. This observation gives rise to the following procedure for generating Z_i's: Pick a random z_i. The probability that $Z_i = g_i^z$ is also a generator is equal to p/q (or approximately $1/2$). Thus, on average, only two tries are needed to ensure that a Z_i is also a generator, and hence a constant number of tries overall to find appropriate z_i's, for $i = 1, 2, 3$.

B. Registration

While our main goal is to provide users with a rewarding scheme for the data they provide, we need to make sure that they cannot double-spend a reward token. A simple solution to this problem is to embed the user ID in the token which will be revealed only when a user tries to double-spend. During registration, a call to Register will create the necessary evidence that can be used to detect double-spending, while a call to GetToken, will equip the user with an initial token that can be used to accumulate rewards. These functions are described in detail below.

$\mathsf{Register}(\mathcal{U}, \mathcal{P})$ is executed between a user and the provider. The user generates a random number $ID_u \in Z_p$ and sends $I = Z_2^{ID_u} \mod q$ to the provider, where Z_2 is a generator of G as per the guidelines mentioned above. I is the users' identifying information while ID_u is kept secret. The uniqueness of I is essential as it will allow the provider to identify the user in case of double-spending. Hence it is stored in the provider's database along with other information about the user.

$\mathsf{GetToken}(\mathcal{U}(ID_u), \mathcal{P})$ is then executed to provide an accumulator token with initial value zero. First, the user's secret ID will be embedded in the token id. To this respect, the user picks random $s, x_1, x_2 \in Z_q$ and forms the quantities $t_1 = g_1^{sID_u} g_2^s$ and $t_2 = g_1^{x_1} g_2^{x_2}$. Then it computes $t = H(t_1, t_2)$, where H is a secure hash function. This value of t will be the token ID to be embedded in the token. However, in order to avoid association with I, it needs to be masked with a random number r. This is done by having the user send to the provider a commitment of the form $C = g^r Z_1^t$. The structure of t will become clear in the next section, when the user tries to update the value aggregated in the token. There, the user will have to release t and prove in zero knowledge the correct form of t_1 and t_2. This is inspired by Brands' scheme [20]; an attempt of using the same token more than once will result in revealing the user ID_u captured by t_1.

However, as both I and C will be embedded in the signature for the initial token, the user needs to convince the provider that both are well formed. To this effect, along with I and C it sends to the provider a proof

$$\pi_0 = PK\{(r, t, ID_u) : C = g^r Z_1^t \wedge I = Z_2^{ID_u}\}$$

of knowledge of the values r, t and ID_u. This proof can be found in the Appendix.

If the proof verifies, the provider creates a signature σ_0 on the commitments C and I as follows. It first generates the values $(a, A_1, A_2, A_3, b, B_1, B_2, B_3)$ as described in Section III-C3. Then, it sets $c = a^x (C{\cdot}I)^{\alpha xy}$ and sends σ_0 to the user. Notice here that while the user provides two public values $C = g^r Z_1^t$ and $I = Z_2^{ID_u}$, the provider signs a tuple of the form $\langle r, t, ID_u, 0\rangle$. Thus the message blocks (m_0, m_1, m_2, m_3) in the description of the signature correspond to the tuple $\langle r, t, ID_u, 0\rangle$. To verify the signature, the user obtains the public key of the provider and applies the tests described in Section III-C3. Most importantly the user needs to test whether c has been formed correctly, that is whether

$$e(X,a)\cdot e(X,b)^r\cdot e(X,B_1)^t\cdot e(X,B_2)^{ID_u}\stackrel{?}{=}e(g,c). \tag{1}$$

In what follows we prove that this is indeed the case.

$$\begin{aligned} LHS &= e(X,a)\cdot e(X,b)^r\cdot e(X,B_1)^t\cdot e(X,B_2)^{ID_u}\\ &= e(g,a)^x\cdot e(g,a)^{xyr}\cdot e(g,a)^{xyz_1t}\cdot e(g,a)^{xyz_2ID_u}\\ &= e(g,a)^{x+xyr+xyz_1t+xyz_2ID_u}\\ &= e(g,a^{x+xyr+xyz_1t_xyz_2ID_u})\\ &= e(g,a^{x+xyr}\cdot A_1^{xyt}\cdot A_2^{xyID_u}) \end{aligned} \tag{2}$$

Similarly,

$$\begin{aligned} RHS &= e(g,c)\\ &= e(g,a^x\cdot(C\cdot I\cdot Z_3^0)^{\alpha xy})\\ &= e(g,a^x\cdot(g^rZ_1^tZ_2^{ID_u})^{\alpha xy})\\ &= e(g,a^x\cdot(g^{\alpha r}Z_1^{\alpha t}Z_2^{\alpha ID_u})^{xy})\\ &= e(g,a^x\cdot(a^ra^{z_1t}a^{z_2ID_u})^{xy})\\ &= e(g,a^x\cdot(a^rA_1^tA_2^{ID_u})^{xy})\\ &= e(g,a^{x+xyr}\cdot A_1^{xyt}\cdot A_2^{xyID_u}) \end{aligned} \tag{3}$$

Thus indeed, the user has a valid signature σ_0 on the token $\tau_0=\langle r,t,ID_u,0\rangle$, with initial value zero. At this point, $\mathsf{GetToken}(\mathcal{U}(ID_u),\mathcal{P})$ returns to the user the pair (τ_0,σ_0) which the user stores for future use. The provider cannot trace the token as the token ID t has never been revealed during the signature generation process.

C. Accumulating rewards

Now, consider a user that submitted data to the provider and expects a reward of value v. In what follows we explain how the existing token of the user can be *updated* to reflect the additional value v. Thus the user does not have to maintain a collection of different tokens but only one as rewards can be *accumulated* into it. However, we need to make sure that this operation happens in a privacy-preserving manner. For the discussion that follows, let $\hat{\tau}=\langle\hat{r},\hat{t},\widehat{ID}_u,\hat{v}\rangle$ be the current token held by the user and $\sigma=(a,\{A_i\},b,\{B_i\},c)$, $i=1,2,3$, its associated signature. A call to $\mathsf{AccumulateToken}(\mathcal{U}(\hat{\tau}),v,\mathcal{P})$ works as follows.

The user picks new, random $r,s,x_1,x_2\in Z_q$ and forms the quantities $t_1=g_1^{s\widehat{ID}_u}g_2^s$ and $t_2=g_1^{x_1}g_2^{x_2}$. Then it sets $t=H(t_1,t_2)$ and constructs a *new* commitment $C=g^rZ_1^tZ_2^{\widehat{ID}_u}Z_3^{\hat{v}}$ on the current values $\widehat{ID}_u$ and $\hat{v}$ with t being the new, updated token ID. Then it sends C along with the old token ID $\hat{t}$ (more precisely it sends $\hat{t}_1,\hat{t}_2$ which make up $\hat{t}$) to the provider. The user will have to prove (i) that it possesses a valid signature σ on the values $(\hat{r},\hat{t},\widehat{ID}_u,\hat{v})$ of the current token, and (ii) that $\hat{t}$ has the right representation. To this respect, it first creates a blinded version of σ. In particular, the user picks random $k,\tilde{k}\in Z_p$ and forms $\hat{\sigma}=(\hat{a},\{\hat{A}_i\},\hat{b},\{\hat{B}_i\},\hat{c})$ as follows [18]:

1) $\hat{a}=a^k$, $\hat{A}_i=A_i^k$, $\hat{b}=b^k$, $\hat{B}_i=B_i^k$, $\hat{c}=c^k$.
   Furthermore, it blinds $\hat{c}$ to obtain a value $\tilde{c}$ that is distributed independently of everything else: $\tilde{c}=\hat{c}^{\tilde{k}}$.
   Then it sends $(\hat{a},\{\hat{A}_i\},\hat{b},\{\hat{B}_i\},\tilde{c})$ to the provider.
2) Let $\delta_a=e(X,\hat{a})$, $\delta_b=e(X,\hat{b})$, $\delta_{B_i}=e(X,\hat{B}_i)$, for $i=1,2,3$ and $\delta_c=e(g,\tilde{c})$.

The user must then carry out the following zero-knowledge proof protocol:

$$PK\{(\hat{r},\hat{t},\widehat{ID}_u,\hat{v},k):\ (\delta_c)^k=\delta_a(\delta_b)^{\hat{r}}(\delta_{B_1})^{\hat{t}}(\delta_{B_2})^{\widehat{ID}_u}(\delta_{B_3})^{\hat{v}}\}. \tag{4}$$

Notice that if the above equation is true, the user indeed possesses a valid signature on the values $(\hat{r},\hat{t},\widehat{ID}_u,\hat{v})$. To see why notice that

$$\begin{aligned} (\delta_c)^k &= \delta_a(\delta_b)^{\hat{r}}(\delta_{B_1})^{\hat{t}}(\delta_{B_2})^{\widehat{ID}_u}(\delta_{B_3})^{\hat{v}} \quad \text{or}\\ e(g,\tilde{c})^k &= e(X,\hat{a})e(X,\hat{b})^{\hat{r}}e(X,\hat{B}_1)^{\hat{t}}e(X,\hat{B}_2)^{\widehat{ID}_u}e(X,\hat{B}_3)^{\hat{v}}\\ e(g,\tilde{c}^k) &= e(X,\hat{a})e(X,\hat{b})^{\hat{r}}e(X,\hat{B}_1)^{\hat{t}}e(X,\hat{B}_2)^{\widehat{ID}_u}e(X,\hat{B}_3)^{\hat{v}} \end{aligned}$$

which satisfies the verification equation of the blinded signature on the token values $(\hat{r},\hat{t},\widehat{ID}_u,\hat{v})$. However, proof (4) alone does not guarantee that the new commitment C is bound to the values $\widehat{ID}_u$ and $\hat{v}$ captured by the signature, so we need to

User | **Provider**

User private input: Current token $\hat{\tau} = \langle \hat{r}, \hat{t}, \widehat{ID}_u, \hat{v} \rangle$ and signature $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$ on $\hat{\tau}$.

Common input: New value v to be added to existing value $\hat{v}$.

$\triangleright$ Construct commitment on new token ID t, user ID $\widehat{ID}_u$ and token value $\hat{v}$.
Pick new, random $r, s, x_1, x_2 \in Z_q$ and form $t_1 = g_1^{s\widehat{ID}_u} g_2^s$ and $t_2 = g_1^{x_1} g_2^{x_2}$. Set $t = H(t_1, t_2)$ as the new token ID.
Construct commitment $C = g^r Z_1^t Z_2^{\widehat{ID}_u} Z_3^{\hat{v}}$.
$\triangleright$ Construct blinded version $\hat{\sigma}$ of signature σ.
Pick random $k, \tilde{k} \in Z_p$. Set $\hat{a} = a^k$, $\hat{A}_i = A_i^k$, $\hat{b} = b^k$, $\hat{B}_i = B_i^k$, $\hat{c} = c^k$ and $\tilde{c} = \hat{c}^{\tilde{k}}$. Set $\hat{\sigma} = (\hat{a}, \{\hat{A}_i\}, \hat{b}, \{\hat{B}_i\}, \tilde{c})$. Let $\delta_a = e(X, \hat{a})$, $\delta_b = e(X, \hat{b})$, $\delta_{B_i} = e(X, \hat{B}_i)$ for $i = 1, 2, 3$ and $\delta_c = e(g, \tilde{c})$.
Construct proof π_1

$C,\ \hat{\sigma},\ \pi_1,\ \hat{t}_1,\ \hat{t}_2 \longrightarrow$

Check that $\hat{t}$ has not been used before.
Check validity of π_1 and $\hat{\sigma}$.
$\triangleright$ Construct new signature σ'.
Generate new values (a', A_i', b', B_i').
$\sigma' = (a', \{A_i'\}, b', \{B_i'\}, c')$ where $c' = a'^x (C Z_3^v)^{\alpha x y}$.

$\longleftarrow \sigma'$

Validate signature σ'.
New token τ' on value $\hat{v} + v$ and signature σ'.

Fig. 3. Reward accumulation

link these two together in a more accurate proof. Additionally, the user has to prove knowledge of the correct representation $\hat{t}_1, \hat{t}_2$ of the token ID $\hat{t}$. This is taken care by proof π_1 below:

$$\begin{aligned} \pi_1 = PK\{(r, t, \hat{r}, \hat{t}, \widehat{ID}_u, \hat{v}, k, \hat{s}, x_1, x_2) : \\ C = g^r Z_1^t Z_2^{\widehat{ID}_u} Z_3^{\hat{v}} \wedge \\ (\delta_c)^k = \delta_a (\delta_b)^{\hat{r}} (\delta_{B_1})^{\hat{t}} (\delta_{B_2})^{\widehat{ID}_u} (\delta_{B_3})^{\hat{v}} \wedge \\ \hat{t}_1 = g_1^{\hat{s}\widehat{ID}_u} g_2^{\hat{s}} \wedge \hat{t}_2 = g_1^{x_1} g_2^{x_2} \} \end{aligned} \tag{5}$$

Thus the user must convince the provider about the validity of π_1 (this proof is deferred to the Appendix) as well as show that (i) the $\hat{A}_i$'s were formed correctly: $e(\hat{a}, Z_i) = e(g, \hat{A}_i)$, and (ii) b and $\hat{B}_i$'s were formed correctly: $e(\hat{a}, Y) = e(g, \hat{b})$ and $e(\hat{A}_i, Y) = e(g, \hat{B}_i)$.

If all these tests succeed, the provider checks if the old token ID $\hat{t}$ has not been used before (provider always records $\hat{t}, \hat{t}_1$ and $\hat{t}_2$ and calls CheckDoubleSpending), and then proceeds to sign the commitment $C = g^r Z_1^t Z_2^{\widehat{ID}_u} Z_3^{\hat{v}}$, where t will be the new token ID and $\hat{v}$ is the current token value.

To this respect, it first generates new values (a', A_i', b', B_i'). Then, it sets c' equal to $c' = a'^x (C Z_3^v)^{\alpha x y}$. Notice here that while the user provides a commitment to the values $(t, \widehat{ID}_u, \hat{v})$, the provider signs a tuple of the form $(t, \widehat{ID}_u, \hat{v} + v)$. Thus, essentially the user obtains a signature on the new token value. To verify the signature, the user obtains the public key of the provider and applies the tests described in Section III-C3. Most importantly it needs to test whether c' has been formed correctly, that is whether

$$e(X, a') \cdot e(X, b')^r \cdot e(X, B_1')^t \cdot e(X, B_2)^{\hat{ID}_u} \cdot e(X, B_3')^{\hat{v}+v} \stackrel{?}{=} e(g, c'). \tag{6}$$

This test is similar to the one described in the end of the registration phase and is omitted here. If the test succeeds, the user stores (τ, σ') as the new token/signature pair, where $\tau = \langle r, t, \hat{ID}_u, \hat{v} + v \rangle$.

CheckDoubleSpending makes sure that the old token cannot be used anymore as the provider keeps a record of the old token ID $\hat{t}$ and the values $\hat{t}_1, \hat{t}_2$ used in its representation. Notice that in proof π_1 (Appendix) the user proves that it knows the

representation of $\hat{t}_1, \hat{t}_2$ by sending $z_1 = e \cdot (\hat{s}\widehat{ID}_u) + x_1$ and $z_2 = e \cdot (\hat{s}) + x_2$ satisfying $g_1^{z_1} g_2^{z_2} = (\hat{t}_1)^e \hat{t}_2$, for some provider's challenge e. If the user attempts to re-use the same token, it has to send new z_1', z_2' for some new challenge e'. Then similar to Brands' scheme [20], the ID of the user can be recovered by solving this system of two equations with two unknowns $(\hat{s}, \widehat{ID}_u)$. Hence double-spending is prevented. All the above is summarized in Figure 3.

D. Redeeming the accumulated rewards

To fully redeem the points or the rewards accumulated in the token thus far, the user must first prove knowledge of the value v claimed to be stored in the token and then obtain a new token of value zero. The case for partial token redemption can be easily handled by slightly modifying AccumulateToken as explained in Section VI. Full redemption is a more interesting case as it requires steps which are a mix of both GetToken and AccumulateToken. Let $\hat{\tau} = \langle \hat{r}, \hat{t}, \widehat{ID}_u, \hat{v} \rangle$ be the current token held by the user and $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$, $i = 1, 2, 3$, its associated signature. A call to RedeemToken$(\mathcal{U}(\hat{\tau}), \mathcal{P})$ works as follows.

The user sends its public identifier $I = Z_2^{\widehat{ID}_u}$ in order to be rewarded for the points $\hat{v}$ collected in the token. Then it picks new, random $r, s, x_1, x_2 \in Z_q$ and forms the quantities $t_1 = g_1^{s\widehat{ID}_u} g_2^s$ and $t_2 = g_1^{x_1} g_2^{x_2}$. The new token ID will be equal to $t = H(t_1, t_2)$. This will be embedded in a new commitment $C = g^r Z_1^t$ which is sent to the provider along with the old token ID $\hat{t}$. The user will have to prove that (i) C and I have the correct representation, (ii) it possesses a valid signature σ on the values $(\hat{r}, \hat{t}, \widehat{ID}_u, \hat{v})$ of the current token, and (iii) that $\hat{t}$ has the right representation. To this respect, it creates a blinded version $\hat{\sigma}$ of the signature as in AccumulateToken and carries out the following zero-knowledge proof:

$$\begin{aligned} \pi_2 = PK\{(r, t, \hat{r}, \hat{t}, \widehat{ID}_u, k, \hat{s}, x_1, x_2) : \\ C = g^r Z_1^t \ \wedge I = Z_2^{\widehat{ID}_u} \ \wedge \\ (\delta_c)^k = \delta_a (\delta_b)^{\hat{r}} (\delta_{B_1})^{\hat{t}} (\delta_{B_2})^{\widehat{ID}_u} (\delta_{B_3})^{\hat{v}} \wedge \\ \hat{t}_1 = g_1^{\hat{s}\widehat{ID}_u} g_2^{\hat{s}} \wedge \hat{t}_2 = g_1^{x_1} g_2^{x_2}\} \end{aligned} \tag{7}$$

Thus the user must convince the provider about the validity of π_2 (this proof is very similar to π_1 and is omitted) as well as show that the blinded signature was formed correctly.

If all these tests succeed, the provider checks if the old token ID $\hat{t}$ has not been used before using CheckDoubleSpending and the values committed in the proof. Then it proceeds to create a new token of value *zero* by signing the commitments C and I as in the case of GetToken.

To this end, the provider first generates the values $(a, A_1, A_2, A_3, b, B_1, B_2, B_3)$ as described in Section III-C3. Then, it sets $c = a^x (C \cdot I)^{\alpha x y}$ and sends the new signature to the user. If the signature verification succeeds, the user ends up with a new token-signature pair that has been initialized to zero value.

V. Analysis

In this section we analyze the security and efficiency aspects of our proposal.

A. Security

When a user tries to accumulate rewards to its token, the token ID cannot be used to track the user in subsequent requests as it is hidden in the commitment C. Additionally, the signature that comes with the token is also blinded during the accumulation phase, so the provider cannot link a user to its previous requests. Thus, intuitively, the provider will not be able to tell which user used the token. However, as mentioned in Section III-A, there exist other side channels that can be used to undermine user privacy, the most important one requiring that the actual connection through which the user interacts with the provider remains anonymous [21].

In what follows, we formally prove that the reward mechanism is privacy preserving as per the Definition 1 in Section III-A. According to the privacy game defined in the same section, our goal is to show that the adversarial provider cannot link a call to AccumulateToken to one of the two users even if all interactions in the pre-challenge and post-challenge phases have been controlled by the adversary. The game involves only AccumulateToken as the identity of the user in all other transactions is known.

Theorem 1: The reward accumulation system is privacy preserving according to Definition 1.

Proof: Let $\mathcal{A}$ be an honest-but-curious adversary and $\mathcal{U}_1, \mathcal{U}_2$ are the two users selected in the privacy experiment shown in Figure 2. Our goal is to show that the probability of $\mathcal{A}$ winning the game is 1/2. This would prove that calls to AccumulateToken are completely indistinguishable.

We will show that $\mathcal{A}$ cannot do better than random guessing in winning the game by showing, using a simulation argument, that the transcript of the interaction produced with the user oracle is indistinguishable from an interaction produced by a simulator $\mathcal{S}$ which does not know the values committed in C.

During the challenge phase, the simulator picks random C and sends it to the adversary. Upon receiving from $\mathcal{A}$ the random challenge e required in the zero-knowledge proof π_1, $\mathcal{S}$ picks random $z_r, z_t, z_{\hat{r}}, z_{\hat{t}}, z_{\widehat{ID}_u}, z_{\hat{v}}, z_k, z_{\hat{s}}, z_1, z_2$ and sets

$$\mu_1 = C^e g^{z_r} Z_1^{z_t} Z_2^{z_{\widehat{ID}_u}} Z_3^{z_{\hat{v}}}$$
$$\mu_2 = (\delta_a)^e (\delta_c)^{z_k} (\delta_b)^{-z_{\hat{r}}} (\delta_{B_1})^{-z_{\hat{t}}} (\delta_{B_2})^{-z_{\widehat{ID}_u}} (\delta_{B_3})^{-z_{\hat{v}}}$$
$$\hat{t}_2 = g_1^{z_1} g_2^{z_2} (\hat{t}_1)^{-e}$$
$$\hat{t} = H(\hat{t}_1, \hat{t}_2)$$

These values are randomly distributed (bear the same distribution as the actual transcript) since they are based on random quantities as in the original transcript of proof π_1. Additionally, they satisfy the verification equation of the proof. Since for any user $\mathcal{U}_b$, there exist random numbers that map it to the transcript of of the above protocol, we conclude that b cannot be determined and that the probability that $\mathcal{A}$ wins the game is exactly 1/2. □

B. Performance aspects

We will analyze the efficiency of the various operations by resorting to the findings of Grewal *et al.* [22] who investigated the efficient computation of optimal-Ate pairing over Barreto-Naehrig curves. The authors implemented pairing computations for different security levels in software for ARM-based platforms such as the ones used in modern day smartphones and tablets. Among others, they developed software for iPad 2 (Apple A5) which uses an ARMv7 Cortex-A9 MPCore processor operating at 1.0 GHz clock frequency and for a Samsung Galaxy Nexus which uses a 1.2 GHz TI OMAP 4460 ARM Cortex-A9 processor. The timings for a C implementation of a BN254 curve (128-bit security level) suggest that a full exponentiation takes between 5.48ms and 4.38ms for the two architectures, while a pairing computation is between 13.82ms and 11.24ms, respectively. In the analysis that follows we will consider the same 128-bit security level along with an upper bound of 5.5ms for the cost of an exponentiation and 14ms for a pairing operation.

Our focus will be on AccumulateToken since it is the most repeated operation (Register and RedeemToken take time only once) and perhaps the most time-consuming one. In the following, we consider both user and provider sides, ignoring the time to generate random quantities or the time to add/multiply numbers as these operations are in the order of a few microseconds according to [22].

- The user makes 4 exponentiations for t_1 and t_2, 4 more for the commitment C, and 9 more to create the blinded signature $\hat{\sigma}$, for a total of 17 exponentiations. Additionally, 6 pairings are required for the computation of the δ quantities.
  In proof π_1, the user needs 4 exponentiations to compute μ_1, and 5 more to compute μ_2.
  If the proof checks out, the user must verify the signature issued on the new token by the provider which requires 20 pairing computations.
  Overall, the time required on the user side is bounded by 466ms.
- The provider must check the proof π_1 and create the new token signature. The amount of work in the proof consists of 14 exponentiations and no pairing operations. On the other hand, generating the signature requires 11 exponentiations.
  Overall, the time required on the provider side is bounded by 138ms. This time can easily be made 2 or 3 times smaller assuming the use of more powerful machines which is a standard practice in server applications than the 1Ghz processors used in this analysis. This further allows the provider to serve multiple user requests.

Working in a similar manner, the timings for the rest of the operations are summarized in Table I. Although the times are more expensive on the user side, a half second delay is barely noticeable as the user is already engaged in data submission for the underlying participatory sensing application.

TABLE I
TIMINGS OF MAIN OPERATIONS. 'E' = # EXPONENTIATIONS, 'P' = # PAIRINGS, TIME IN MILLISECONDS.

	User			***Provider***		
Operation	E	P	Time(ms)	E	P	Time(ms)
Register	1	-	5.5	-	-	-
GetToken	9	20	330	15	-	83
AccumulateToken	26	26	507	25	-	138
RedeemToken	23	24	463	25	-	138

In terms of *storage*, the provider has to maintain a collection of tokens to check against double-spending attempts. For each token, it has to store 4 128-bit numbers: t_1 and t_2 which give rise to the token ID t, and z_1, z_2 used in the zero-knowledge

proof. Thus the overhead for the provider is minimal, given also the fact that the sensed data collected along with these tokens are typically orders of magnitude larger. This overhead can be reduced further by introducing an expiration day and delete expired tokens as suggested in Section VI.

At any time, the user has to maintain only *one* reward token $\tau = \langle r, t, ID_u, v\rangle$ and its associated signature $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$, as rewards can be accumulated. The total memory overhead does not exceed 2 Kbytes.

In terms of *communication*, all protocols require two messages (the ZK proofs can be made non-interactive using the Fiat-Shamir heuristic) with the more 'expensive' one being the accumulation protocol. In the first message (recall Figure 3), the user has to send the new commitment C, the token numbers $\hat{t}_1$, $\hat{t}_2$, the proof π_1 (z values along with μ_1, μ_2) and the blinded signature $\hat{\sigma}$ (δ quantities). This amounts to a total of $21 \times 128 = 2688$ bytes.

The provider has to reply back with a new signature $\sigma' = (a', A'_i, b', B'_i, c)$ which results in another $9 \times 128 = 1152$ bytes communicated.

Overall, the communication overhead is minimal as transmitting these quantities over modern 3G/4G networks is barely noticeable by the user. This demonstrates the feasibility of our approach in all aspects.

VI. Discussion

Here we consider a few alternatives that can be used to enhance the system further.

A. Redeeming only a portion of the token amount

In the previous sections, we showed how awards can be accumulated together in a single token. But what happens if a user wants to redeem a *portion* of the amount stored in the token? The implicit assumption here was that the full amount has to be redeemed, in which case the user has to obtain a new token of value zero.

It is straightforward, however, to expand the proposed system in order to allow only part of the token value to be redeemed. This can be done by introducing a new function RedeemPartialToken which is basically a variant of AccumulateToken:

- RedeemPartialToken$(\mathcal{U}(\tau), v, \mathcal{P})$ is a protocol executed between a user possessing a signed reward token τ and $\mathcal{P}$. Upon successful execution, the value of the token is decreased by the amount v, provided the current token value is it least as large as v.

The operation of this function is entirely similar to that of AccumulateToken. The only extra requirement is that in proof π_1, the user must also prove that it has enough funds in the token to spend (basically the current accumulated value is bigger than v). This addition is underlined in the proof below:

$$\begin{aligned}\pi'_1 = PK\{(r, t, \hat{r}, \hat{t}, \widehat{ID}_u, \hat{v}, k, \hat{s}, x_1, x_2) : \\ C = g^r Z_1^t Z_2^{\widehat{ID}_u} Z_3^{\hat{v}} \ \wedge \underline{\hat{v} \geq v} \wedge \\ (\delta_c)^k = \delta_a (\delta_b)^{\hat{r}} (\delta_{B_1})^{\hat{t}} (\delta_{B_2})^{\widehat{ID}_u} (\delta_{B_3})^{\hat{v}} \wedge \\ \hat{t}_1 = g_1^{\hat{s}\widehat{ID}_u} g_2^{\hat{s}} \wedge \hat{t}_2 = g_1^{x_1} g_2^{x_2}\}\end{aligned} \tag{8}$$

If the proof verifies, the provider proceeds to sign a quantity of the form $(t, \widehat{ID}_u, \hat{v} - v)$, thus essentially removing v credits from the token. To this end, the provider first generates the values $(a, A_1, A_2, A_3, b, B_1, B_2, B_3)$ and then it sets $c = a^x (C \cdot Z_3^{-v})^{\alpha x y}$. If the signature verification succeeds, the user ends up with a new token of value $\hat{v} - v$.

B. Simplifying double-spending detection

The mechanism we used to detect and prevent double-spending involved embedding the user ID in the token ID. Hence an attempt to re-use a token with the same ID resulted in identifying the user.

If we are interesting only in detecting double-spending without penalizing cheating users, the scheme can be simplified. In particular, a token will consist of a tuple $\langle rnd, tID, val\rangle$ without the need to maintain the secret user identity ID_u. Now the token ID can just be a random number that changes with every token accumulation without a reference to ID_u.

The signature of the token will involve only the three elements $\langle rnd, tID, val\rangle$, reducing the public key size and thus saving signature generation and verification times further. Finally, the proof π_1 in AccumulateToken would also simplify to

$$\begin{aligned}\pi_1 = PK\{(r, t, \hat{r}, \hat{t}, \hat{v}, k) : \ C = g^r Z_1^t Z_2^{\hat{v}} \wedge \\ (\delta_c)^k = \delta_a (\delta_b)^{\hat{r}} (\delta_{B_1})^{\hat{t}} (\delta_{B_2})^{\hat{v}}\}\end{aligned}$$

and CheckDoubleSpending would simply involve searching whether the current token ID exists in the provider's database.

C. Minimizing information retention

In order for double spending detection to be possible in the basic system, the provider needs to maintain a database keeping track of all token IDs used thus far. However, in order to avoid storing these token numbers forever, we can introduce some kind of token expiration date after which the token ID can be deleted from the database. This could be done either by having the expiration date encoded as an attribute in the tokens (so the user would have to prove that the token is still valid during spending) or by changing the provider's public key (e.g. once a year) and rendering older tokens invalid.

VII. Conclusions

Existing participatory sensing systems allow users to report sensed data to a campaign administrator. However, a drawback of existing solutions is that they do not allow users to get rewards for the data they provide, thus lacking the incentives to increase user participation.

In this work, we have proposed a rewarding mechanism that enables users to anonymously interact with the service provider when submitting detailed sensory data by accumulating rewards of different sessions into a single token. This increases usability while preventing providers to track users by the tokens they carry or spend. We have formally analyzed the security of our proposal and evaluated its performance in realistic deployment settings. Our results suggest that our scheme incurs only minor performance overhead on both provider and users. We therefore hope that our findings will motivate further research in this area.

References

[1] B. Guo, Z. Yu, X. Zhou, D. Zhang. "From participatory sensing to mobile crowd sensing." In Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014.

[2] B. Hoh, T. Yan, D. Ganesan, K. Tracton, T. Iwuchukwu, and J.-S. Lee, "Trucentive: A game-theoretic incentive platform for trustworthy mobile crowdsourcing parking services." In The 15th IEEE Conference on Intelligent Transportation Systems (ITSC), 2012.

[3] J.-S. Lee and B. Hoh. "Sell your experiences: a market mechanism based incentive for participatory sensing." In IEEE International Conference on Pervasive Computing and Communications (PerCom), 2010.

[4] D. Yang, G. Xue, X. Fang, and J. Tang. "Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing." In Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (Mobicom 2012), Aug. 2012, pp. 173-184.

[5] K. Shilton. "Four Billion Little Brothers?: Privacy, Mobile Phones, and Ubiquitous Data Collection." In Communications of the ACM 52 (11) (2009) 48-53.

[6] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, V. D. Blondel. "Unique in the Crowd: The Privacy Bounds of Human Mobility." Scientific reports 3, Article number: 1376 (2013)

[7] D. Christin. "Privacy in mobile participatory sensing: Current trends and future challenges." Journal of Systems and Software 116 (2016): 57–68.

[8] Francesco Restuccia, Sajal K. Das, and Jamie Payton. "Incentive mechanisms for participatory sensing: Survey and research challenges." In ACM Transactions on Sensor Networks (TOSN), May 2016.

[9] S. Reddy, D. Estrin, M. Hansen, and M. Srivastava. "Examining Micropayments for Participatory Sensing Data Collections." In Proceedings of the 2010 ACM International Conference on Ubiquitous Computing (Ubicomp '10).

[10] Q. Li and G. Cao. "Providing efficient privacy-aware incentives for mobile sensing." In IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014.

[11] T. Dimitriou and I. Krontiris. "Privacy-Respecting Auctions as Incentive Mechanisms in Mobile Crowd Sensing.". In the 9th IFIP International Conference on Information Security Theory and Practice, 2015.

[12] D. Christin, C. Roßkopf, M. Hollick, L.A. Martucci, S.S. Kanhere. "Incognisense: An anonymity-preserving reputation framework for participatory sensing applications." In Pervasive and mobile Computing, Jun 30;9(3):353-71, 2013.

[13] A. Rupp, F. Baldimtsi, G. Hinterwlder, and C. Paar. "Cryptographic theory meets practice: Efficient and privacy-preserving payments for public transport." In ACM Trans. Inf. Syst. Secur., 17(3):10:110:31, 2015.

[14] Alberto Blanco-Justicia and Josep Domingo-Ferrer. "Privacy-preserving Loyalty Programs." in Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance - 9th International Workshop, DPM 2014.

[15] Z. Yang, S. Yu, W. Lou, and C. Liu. "P^2: Privacy-preserving communication and precise reward architecture for V2G networks in smart grid." IEEE Trans. Smart Grid, 2(4):697706, 2011.

[16] R. Dingledine, N. Mathewson, P. Syverson, "Tor: the second-generation onion router," In Proceedings of the 13th Conference on USENIX Security Symposium (USENIX Security), pp. 21-38, 2004.

[17] T. P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing." In CRYPTO, 1991.

[18] J. Camenisch and A. Lysyanskaya. "Signature Schemes and Anonymous Credentials from Bilinear Maps." In CRYPTO, volume 3152 of Lecture Notes in Computer Science, pages 56–72. Springer, 2004.

[19] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[20] Stefan Brands. "An efficient off-line electronic cash system based on the representation problem." Technical Report CS-R9323, CWI, 1993.

[21] I. Krontiris, F.C. Freiling, T. Dimitriou. "Location privacy in urban sensing networks: research challenges and directions." In IEEE Wireless Communications, 2010.

[22] G. Grewal, R. Azarderakhsh, P. Longa, S. Hu, and D. Jao. "Efficient implementation of bilinear pairings on ARM processors." In International Conference on Selected Areas in Cryptography, 2012.

APPENDIX

A. Proof π_0

$$\pi_0 = PK\{(r, t, ID_u) : C = g^r Z_1^t \wedge I = Z_2^{ID_u}\}$$

To prove knowledge of the values r, t and ID_u, the prover and verifier engage in the following steps:

1) Prover picks random ρ, τ, ι.
2) Prover computes
$$\mu_1 = g^{\rho} Z_1^{\tau}$$
$$\mu_2 = Z_2^{\iota}$$
3) Verifier sends challenge e.
4) Prover sets
$$z_r = \rho - e \cdot r$$
$$z_t = \tau - e \cdot t$$
$$z_{ID_u} = \iota - e \cdot ID_u$$
and sends these values, along with μ_1, μ_2, to the verifier.
5) The verifier accepts the proof if and only if
$$\mu_1 \stackrel{?}{=} C^e g^{z_r} Z_1^{z_t}$$
$$\mu_2 \stackrel{?}{=} I^e Z_2^{z_{ID_u}}$$

B. *Proof* π_1

Consider the following proof between a prover and a verifier:

$$\pi_1 = PK\{(r, t, \hat{r}, \hat{t}, \widehat{ID}_u, \hat{v}, k, \hat{s}, x_1, x_2) :$$
$$C = g^r Z_1^t Z_2^{\widehat{ID}_u} Z_3^{\hat{v}} \wedge$$
$$(\delta_c)^k = \delta_a (\delta_b)^{\hat{r}} (\delta_{B_1})^{\hat{t}} (\delta_{B_2})^{\widehat{ID}_u} (\delta_{B_3})^{\hat{v}} \wedge$$
$$\hat{t}_1 = g_1^{\hat{s}\widehat{ID}_u} g_2^{\hat{s}} \wedge \hat{t}_2 = g_1^{x_1} g_2^{x_2}\}$$

To prove knowledge of the quantities $r, t, \hat{r}, \hat{t}, \widehat{ID}_u, \hat{v}, k, \hat{s}$, x_1, x_2, the prover and verifier engage in the following steps:

1) Prover picks random $\rho, \tau, \hat{\rho}, \hat{\tau}, \hat{\iota}, \hat{\nu}, \kappa, \hat{\sigma}, \chi_1, \chi_2$.
2) Prover computes
$$\mu_1 = g^{\rho} Z_1^{\tau} Z_2^{\hat{\iota}} Z_3^{\hat{\nu}}$$
$$\mu_2 = (\delta_c)^{\kappa} (\delta_b)^{-\hat{\rho}} (\delta_{B_1})^{-\hat{\tau}} (\delta_{B_2})^{-\hat{\iota}} (\delta_{B_3})^{-\hat{\nu}}$$
3) Verifier sends challenge e.
4) Prover sets
$$z_r = \rho - e \cdot r$$
$$z_t = \tau - e \cdot t$$
$$z_{\hat{r}} = \hat{\rho} - e \cdot \hat{r}$$
$$z_{\hat{t}} = \hat{\tau} - e \cdot \hat{t}$$
$$z_{\widehat{ID}_u} = \hat{\iota} - e \cdot \widehat{ID}_u$$
$$z_{\hat{v}} = \hat{\nu} - e \cdot \hat{v}$$
$$z_k = \kappa - e \cdot k$$
$$z_{\hat{s}} = \hat{\sigma} - e \cdot \hat{s}$$
$$z_1 = e \cdot (\hat{s}\widehat{ID}_u) + x_1$$
$$z_2 = e \cdot (\hat{s}) + x_2$$
and sends these values, along with μ_1, μ_2, to the verifier.
5) The verifier accepts the proof if and only if
$$\mu_1 \stackrel{?}{=} C^e g^{z_r} Z_1^{z_t} Z_2^{z_{\widehat{ID}_u}} Z_3^{z_{\hat{v}}}$$
$$\mu_2 \stackrel{?}{=} (\delta_a)^e (\delta_c)^{z_k} (\delta_b)^{-z_{\hat{r}}} (\delta_{B_1})^{-z_{\hat{t}}} (\delta_{B_2})^{-z_{\widehat{ID}_u}} (\delta_{B_3})^{-z_{\hat{v}}}$$
$$g_1^{z_1} g_2^{z_2} \stackrel{?}{=} (\hat{t}_1)^e \hat{t}_2$$

Then it stores the values $\hat{t}_1, \hat{t}_2, z_1$ and z_2. If the user attempts to use the same token again, with the same representation $\hat{t}_1, \hat{t}_2$, then the identity of the user will be revealed. Hence any double-spending attempt will be caught.
Notice that both proofs can be made non-interactive by making the challenge e of the verifier equal to the hash of the μ_1, μ_2 values committed by the prover.