Catching Cheats with Interactive Proofs: Privacy-preserving Crowd-sourced Data Collection Without Compromising Integrity

Akshay Dua, Nirupama Bulusu, and Wu-chang Feng
Department of Computer Science
Portland State University
{akshay,nbulusu,wuchang}@cs.pdx.edu

ABSTRACT

Crowd-sourced sensing systems allow people to voluntarily contribute sensor data from mobile devices. They enable numerous applications, including weather and traffic monitoring. However, their proliferation is at risk if the problems of data integrity and privacy persist. People will be reluctant to contribute sensitive information if they cannot trust the system to maintain their privacy, and the system will be reluctant to accept any data transformed to preserve privacy without proof that the transformation was computed accurately. We present an interactive proof protocol that allows an intermediary to convince a data consumer that it is accurately performing a privacy-preserving transformation with inputs from trusted sources, without providing those inputs to the consumer. We provide soundness and correctness proofs for the protocol, discuss its current limitations, and describe its parameters and their effect on data integrity and privacy when tweaked.

1. INTRODUCTION

Integrity of the collected data, and the privacy of data sources are first order concerns for crowd-sourced sensing systems. Such systems can enable critical applications like faster emergency response, and improve health, environment, and traffic monitoring [13, 10, 1]. However, people will be reluctant to volunteer sensitive information (e.g. location, health statistics) if they cannot trust the system to protect their privacy. Conversely, if the volunteered information is first modified to protect privacy, then the system will be reluctant to trust that modification without proof of its integrity.

Consequently, integrity and privacy compete with each other. If the collected data has been previously transformed to preserve privacy (e.g. mixed, aggregated), then a data consumer cannot determine the transformation's integrity unless the raw data used as input is presented as well. However, if the raw data is presented, then the privacy of the data sources gets compromised.

This work attempts to *simultaneously* provide integrity and privacy guarantees for published data without significantly compromising either. The system model assumes a set of trusted data sources that collect and forward sensory data to a *privacy proxy*, which performs a privacy-preserving transformation on the received data, and finally forwards the result to a data consumer. The goal is to assure the data consumer that the proxy indeed computed the expected privacy transformation using data from expected sources (integrity) without providing the consumer with that data (privacy).

Much of the existing work on crowd-sourced sensing, with a focus on integrity and privacy, adheres to this model. Moreover, such a model has the advantage of decoupling the privacy transformation from data collection, enabling transformations that mix data from multiple sources, or perform application-specific data perturbations on the same data. Examples of this model include our earlier work on the design and implementation of a Trusted Sensing Peripheral (TSP) that produces and publishes trustworthy sensory information to a data portal via the user's personal mobile device [5]. The mobile device is allowed to aggregate the data from the TSP before forwarding it. In this case, the mobile device can play the role of the privacy proxy, while the portal plays the role of the data consumer. Other examples include PoolView [6], which introduces the personal privacy firewall to perturb a user's raw data before publishing it to an aggregation service, DietSense [13], which provides private storage to a user where she can edit the images collected from her phone before sharing it further, and AnonySense [11], which uses a trusted server to mix data from at least l clients to provide l-anonymity.

This paper presents an interactive proof protocol [9], using which, only an honest privacy proxy can convince a data consumer that it is correctly computing the expected privacy-preserving transformation, while protecting the privacy of its data sources. The key idea is that unlike traditional interactive proofs with one prover (privacy proxy) and one verifier (data consumer), ours involves a collaboration between the verifier and an additional trusted party (data source) to keep the prover in check. We provide soundness and correctness proofs for the protocol, discuss its current limitations, and describe its parameters and their effect on data integrity and privacy when tweaked.

2. PROBLEM STATEMENT

Only an honest privacy proxy P should be able to convince a data consumer C that it is indeed transforming data $D_j = \{d_{1j}, d_{2j}, ..., d_{nj}\}$, received in interval j, from a set of sources $S = \{s_1, s_2, ..., s_n\}$, using only the transformation function f_{priv}. C receives the result $p_j = f_{priv}(D_j)$, but never the data D_j. The system model is shown in Figure 1.

3. BACKGROUND

Goldwasser et al. [9] introduced the concept of interactive proof systems where a prover P exchanges messages with a verifier V and convinces it with high probability that some

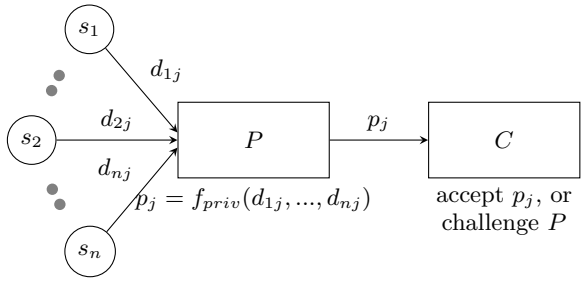


Figure 1: System model

statement is true. As a simple example [2], consider how Alice proves to Bob that she can distinguish between Coke and Pepsi. Alice turns her back, then Bob tosses a fair coin, puts Coke or Pepsi in a glass according to the result, and challenges Alice to say which drink it is. Alice tastes the drink and tells Bob which one it is. If Alice *cannot* actually distinguish between the drinks, then she has a chance of being right with probability 1/2. However, when the experiment is repeated k times (each called a "round") her chances are slimmer: $1/2^k$. Thus, when $k = 10$ and Alice answers correctly each time, Bob will be convinced with probability $1 - 1/2^{10}$ or 99.9% that Alice can taste the difference between the drinks. It is key, that the prover (Alice) not know anything about the challenge before it is presented to her by the verifier (Bob). Otherwise, a malicious prover would answer correctly each time. For example, if Alice secretly held a mirror and saw Bob flip the coin and pour out the respective drink, she would simply have to name that drink once Bob challenges her.

Interactive proof systems must be, a) *complete*: any true statement must be accepted by the verifier, and b) *sound*: any false statement must be rejected by the verifier (barring a negligible probability of error). Based on such a proof system, we have constructed a challenge–response protocol that allows a privacy proxy (the prover) to prove to a data consumer (the verifier) that it is honestly performing the privacy transformation, but *without* giving the consumer the inputs to that transformation.

4. THREAT MODEL

Our protocol is designed to prevent a malicious C from learning about the raw data D from the set of trusted sources S, and prevent a malicious P from either using a different transformation: f'_{priv} instead of f_{priv}, or different data: D' instead of D, or both. Since data consumer C has a vested interest in the integrity of the privacy-preserving transformation, it is assumed to be an adversary of privacy, but not of integrity. The privacy proxy P on the other hand is assumed to be an adversary of integrity, but not of privacy. The reason being, that P may actually be controlled by the sources themselves, or maybe trusted by them to not reveal their data. An implication of the above adversarial model is that P and C do not collude in any way.

We do not consider threats from eavesdropping adversaries since standard network security protocols, like TLS, can easily be used to combat them. We also do not consider situations in which P, C, and S do not interact sincerely, implying that neither side suppresses a response expected by the other. Further, we assume S is trusted, its origin is anonymized via a mix network like Tor [4], it can perform anonymous signatures like those described in group signature schemes [3], and that C honestly executes the protocol (since C has a vested interest in collecting high-integrity data). However, C is free to perform offline privacy attacks on the data received from P. Note that our work focuses on *content*, as opposed to, *origin* integrity and privacy.

5. INTERACTIVE PROOF PROTOCOL

As shown in Figure 1, C collects data, transformed using f_{priv} at regular intervals, from sources S, via P. d_{1j} represents the data sent by source s_1 during interval j. Now, in each interval, C can choose to accept the transformed value p_j as-is, or challenge P to prove p_j's integrity. This challenge message marks the beginning of the interactive proof protocol. In Section 5.5, we discuss our protocol's parameters that allow C to challenge all received transformations, or randomly challenge a portion of them.

For simplicity, we explain the protocol using only one trusted data source, say s_1. Consequently, we assume that data d_{1j} is an m-tuple of raw sensory data values $[x_{1j}, ..., x_{mj}]$ collected in interval j. Now, instead of P computing $p_j = f_{priv}(d_{1j}, ..., d_{nj})$ as shown in Figure 1, it will be computing $p_j = f_{priv}(x_{1j}, ..., x_{mj})$. An example of this single-source scenario is a location-based service, where instead of sending raw GPS coordinates d_{1j} from s_1, P sends a coarser region covered by those coordinates. The generalization to multiple sources will be postponed to a more comprehensive version of this paper.

5.1 Constraints

Our protocol must adhere to the following constraints:

1. The data consumer is never provided with the raw data d_{1j}. This is the basic privacy requirement.
2. The data consumer should be able to determine with *high probability* that:
   - The inputs to the privacy transformation f_{priv} consist of only those coming from s_1.
   - No other transformation but f_{priv} is being performed on those inputs.
3. As implied in Section 3, P must not know the challenge *before* it is presented by C. Otherwise, P may tailor its response to pass only that challenge. The key to an interactive proof is that an honest prover can pass both challenges, but a dishonest one can only pass either by guessing.
4. P must not know *beforehand* the interval j in which it will be challenged. Otherwise, it would compute f_{priv} correctly only during those intervals.

5.2 Protocol Details

For the protocol to work, we require a shared symmetric key k_{sc} between the source s_1 and the data consumer C, a buffer at s_1 that is large enough to store data collected in b distinct intervals, and we need to impose the following constraint on the transformation function f_{priv}: given a function $g(r, x)$ that obfuscates input x using random number r, we require that

$$f_{priv}(g(r, x_{1j}), ..., g(r, x_{mj})) = g(r, f_{priv}(x_{1j}, ..., x_{mj})) \quad (1)$$

s_1	**P**	**C**
$j = 1$: *sense* $x_{11}, ..., x_{m1}$		
save $[x_{11}, ..., x_{m1}], j = 1$		
$s_1 \rightarrow P$: $x_{11}, ..., x_{m1}$	$d_{11} = x_{11}, ..., x_{m1}$	
	$P \rightarrow C$: $p_1 = f_{priv}(d_{11})$	p_1
$j = 2$: *sense* $x_{12}, ..., x_{m2}$		
$s_1 \rightarrow P$: $x_{12}, ..., x_{m2}$	$d_{12} = x_{12}, ..., x_{m2}$	
	$P \rightarrow C$: $p_2 = f_{priv}(d_{12})$	p_2
...	...	...

Table 1: Normal Operation

So for example, let $g(r, x) = r \cdot x$, and $f_{priv}(x_{1j}, ..., x_{mj}) = mean(x_{1j}, ..., x_{mj})$, then:

$$\begin{aligned} f_{priv}(g(r, x_{1j}), ..., g(r, x_{mj})) &= mean(r \cdot x_{1j}, ..., r \cdot x_{mj}) \\ &= r \cdot mean(x_{1j}, ..., x_{mj}) \\ &= g(r, f_{priv}(x_{1j}, ..., x_{mj})) \end{aligned}$$

Section 7 discusses the practicality of the above imposed constraint and highlights examples of privacy-preserving transformations that indeed satisfy the constraint. We now discuss the details of the protocol.

Table 1 shows, that while performing its normal sensing duties, source s_1 continues to randomly pick an interval j and save the data $[x_{1j}, ..., x_{mj}]$ collected in that interval. Once the buffer is full (has b intervals worth of data) s_1 uniformly and randomly selects an interval of data from its buffer (Table 2). It then sends one of two sets of messages to P and C corresponding to each challenge, after which a simple indicator challenge message from C to P begins the interactive proof. Once the proof is complete, s_1 can purge the respective interval of data from its buffer, thus making room for data from the next randomly picked interval. Other notation includes $E_{k_{sc}}$, which is some symmetric encryption scheme using key k_{sc}.

The proof starts when C challenges P about the integrity of some transformed result p_j received in the past (Table 2); we call this p_j a commitment from P. However, which interval j is challenged, is decided by s_1 and selectively revealed to C. The goal is to not let C have j and the obfuscated values M_2 together. Otherwise, it can recover r and therefore the raw values from s_1. What is noteworthy, is that unlike traditional interactive proofs with one prover and one verifier, ours involves a collaboration between the verifier C and an additional trusted party (the data source) to keep the prover P in check.

There are two types of challenges and associated tests that take place. Message M_1 corresponds to the first type of challenge, while M_2 to the second type. The first type is designed to test the integrity of P's past commitment, while the other tests the computation of f_{priv} itself. The padding PAD assures that M_1 and M_2 have the same length. The encryption further ensures that the messages are statistically similar. That way, a malicious P intercepting message M_1 or M_2 will be unable to discern them and hence, not know which type of test will be performed by C.

Intuitively, the protocol is trying to ensure that C does not have r or j, and the obfuscated message M_2 together in any given interval j. Otherwise, C could first recover r using the property in Equation (1), and then recover s_1's raw data values from M_2 (remember that $g(r, x)$ only obfuscates x, and is not a cryptographic one-way function).

At the same time, the protocol wants to ensure that P does not know what type of test is going to be performed by C. Otherwise, P might tailor its response to pass only that test. For example, assume that in each interval, a malicious P fabricates the transformation it sends to C. Also assume, that it caches all raw data values ever received from s_1, and computes and caches (but does not send to C) the correct p_j for each interval as well. Now, suppose it knows that challenge 1 is underway. Then, it computes p as required by the challenge, recovers some r (not necessarily the right one) using $p = g(r, p_j)$ for some interval j, extracts each raw value from the obfuscated message M_0, and compares the set of raw values with those cached for that interval. If a match is found, P knows it has the correct j and therefore r, otherwise it picks another interval j and repeats this process till a match is found. With the correct value of r, it can provide C with the expected response p and pass challenge 1 each time. Now, if P did not know which challenge was underway, then the above attack (called the *find-r* attack) can only pass challenge 1 but not challenge 2. So, with probability 1/2 a malicious P's response to the challenge would be rejected by C. However, an honest P's response could simultaneously pass both tests.

5.3 Analysis

Without going into details, we now point out that our protocol satisfies all constraints mentioned in Section 5.1, but the second one. The second constraint can be satisfied by parametrizing the protocol, and this is discussed in the next section. We now show that our interactive proof is *complete* and *sound*.

Completeness is easier to prove. From Table 2, we can see that any honest P that computes and publishes $f_{priv}(d_{1j})$, and then $f_{priv}(g(r, x_{1j}), ..., g(r, x_{mj}))$ when challenged, will indeed pass either of C's integrity tests with its response p.

We prove soundness by case analysis. But first, we can safely assume that $g(r, x)$ is honestly computed. Since during either challenge, one honest party, C (honest with respect to executing the protocol) or s_1 (trusted), computes g. Thus, if P computes some g' instead of g it will fail either challenge. There are three cases we must consider: P may use fabricated inputs to the transformation f_{priv}, may use the wrong transformation, or both. Throughout, we will refer to notation used in Table 2.

Case 1. Fabricated inputs: If a malicious P publishes $p_j = f_{priv}(d'_{1j})$ instead of $p_j = f_{priv}(d_{1j})$, then during the challenge it must still prove to C that $p = g(r, p_j)$. If it honestly computes p, then this check will fail. If P mounts an attack similar to *find-r* described in Section 5.2, it can pass C's first test, but not the second. Since in the second test, M_2 comes directly from the trusted source s_1.

Case 2. Wrong transformation: If a malicious P is computing $f'_{priv}(d_{1j})$ instead of $f_{priv}(d_{1j})$, then to pass either test, f'_{priv} must share the same relationship with g that f_{priv} does (Equation (1)). Now say that it does, then P could pass C's first test, but again, not the second one. Since in the second test, C itself computes f_{priv}. Also, P's *find-r* attack suffers the same fate.

Case 3. Wrong inputs and transformation: If a malicious P is computing $f'_{priv}(d'_{1j})$ instead of $f_{priv}(d_{1j})$, then the *find-r* attack could pass test 1, but not test 2. Since in test 2, C computes f_{priv} with trusted inputs from

s_1	**P**	**C**
From b saved intervals, randomly *pick* j Then, *pick* random number r Then, with probability 1/2, *either* send: $s_1 \rightarrow C$: $M_1 = E_{k_{sc}}(r, j, PAD)$ $s_1 \rightarrow P$: $M_0 = g(r, x_{1j}), ..., g(r, x_{mj})$ OR send: $s_1 \rightarrow C$: $M_2 = E_{k_{sc}}(g(r, x_{1j}), ..., g(r, x_{mj}))$ $s_1 \rightarrow P$: $M_0 = g(r, x_{1j}), ..., g(r, x_{mj})$		
	M_0 On receiving challenge, $p = f_{priv}(M)$ $P \rightarrow C$: p	M_1 OR M_2 $\leftarrow$ Challenge P p **Test 1** (if received M_1): $r, j, PAD = D_{k_{sc}}(M_1)$ if $p \neq g(r, p_j)$, **reject** **Test 2** (if received M_2): if $p \neq f_{priv}(D_{k_{sc}}(M_2))$, **reject**

Table 2: Interactive proof of integrity for privacy-preserving transformations performed by P

source s_1.

5.4 An Attack on Privacy

It is possible for C to launch an offline privacy attack on the $g(r, x_{1j}), ..., g(r, x_{mj})$ data values it receives during challenge 2. As mentioned before, g is only assumed to be an obfuscation function that can easily be reversed. Leading to ways in which r could be retrieved, and then possibly the raw data values x_{ij}, $1 \leq i \leq m$.

This attack is the same as *find-r* described earlier, but without the cache of raw trusted data values from the source. Without this cache, C will not know if the retrieved raw values are actually correct. It would have to *guess* if the retrieved raw data values seems "plausible". This attack can be mitigated by increasing the number of possible choices from which to guess. We discuss this further in Section 5.5. In the worst case, C *could* obtain raw data values for 1/2 the number of intervals in which it challenges P. Furthermore, the plain-text guessing required in this attack may require human intervention, making this attack more costly.

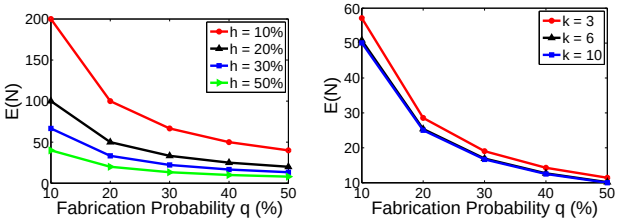
5.5 Parameters

Table 2 describes only one round of the interactive proof. Using multiple rounds, C can gain more confidence in the integrity of the published value p_j being challenged. Further, by challenging more often, C can gain more confidence in the integrity of the data stream in general. Based on such observations, we have defined the following parameters:

- h: the percentage of intervals that C will challenge P. So, if h is 20%, then C will randomly challenge P during one of every five intervals. Note that s_1 must know h as well, since it initiates the challenge. A larger h, will provide more integrity, but will reduce privacy since C will receive more obfuscated values that it can potentially attack offline.

- k: the number of rounds each interactive proof is executed. As mentioned in Section 3, with $k = 10$, C can have 99.9% confidence in the integrity of the published value being challenged. Closely related to k, is an interesting metric we call the *confidence index* of P: defined as the number of challenges P has passed, over the total times challenged. When k is low (say 1, i.e. 50% chance a published value is incorrect), and P's confidence index is high (say 100%) then C can still be sure about the integrity of the already published values from P. However, if P's confidence index is low, then C can choose to increase k and at least gain more confidence each time it challenges P.

- b: the number of intervals of data that s_1 needs to save before initiating challenges. Increasing b can mitigate the privacy attack discussed in the previous section. It will increase a malicious C's ambiguity about the correct inputs to f_{priv} that might have created a given p_j. For a given interval j, C would have to choose between b/h possible sets of inputs. For example, if $b = 1000$ and $h = 1/5$, then C would have to guess the right inputs from 5000 possible ones. In the worst case, C can retrieve $h/2$ intervals of raw data.



(a) Varying h, $k = 1$. (b) Varying k, $h = 20\%$.

Using parameters h and k we have the following equation for the expected number of intervals $E(N)$ before C detects a malicious P. Here, P is fabricating a transformed value p_j with probability q.

$$E(N) = \sum_{n=1}^{\infty} n \times (1 - hq(0.5^k))^{n-1} \times hq(0.5^k) \qquad (2)$$

Plots of $E(N)$ while varying either h or k are shown in Figures 2(a),2(b). When $h = 20\%$, $q = 10\%$ a malicious P is

expected to fail a challenge in 100 intervals, and publish 10 fabricated transformed values. Also, by setting $k = 3$ (instead of $k = 1$) we can reduce time to detection by $\approx 40\%$.

6. RELATED WORK

Much of the previous security oriented work in crowd-sourced sensing has focused *either* on data integrity, or privacy. PoolView [6] enables community statistics to be computed using perturbed private data, but trusts its users to honestly perturb that data. PriSense [14] employs data slicing and mixing to provide privacy while still supporting a wide variety of aggregation functions. However, it is assumed the functions themselves are honestly computed. Our previous work on Trusted Sensing Peripherals [5] supports high-integrity aggregation, but does not provide privacy guarantees.

VPriv [12] makes a strong attempt to offer integrity and privacy for computing tolls over paths driven by vehicles. However, due to the use of an additive homomorphic commitment scheme, VPriv can only guarantee the integrity of additive functions. Additionally, the random spot checks needed to keep drivers honest may compromise privacy.

Fully homomorphic encryption schemes (addition and multiplication operations supported over ciphertexts) could go a long way in solving the integrity and privacy problem. The first such scheme was recently introduced by Gentry et al. [7]. However, computation on ciphertexts is still widely considered to be computationally expensive.

7. LIMITATIONS

It remains to be seen what types of privacy-preserving transformations can work with our protocol, given the constraint in Equation (2). We have shown that a transformation computing the mean of its inputs can be used. Other transformations we plan to investigate include mixing data from multiple sources to provide k-anonymity [15], and possibly location blurring.

Interactive proofs require their participants to be online. Hence, the data sources need to be online while the proof is taking place. Future work could include constructing a non-interactive proof that achieves the same goals.

Interactive proofs can be zero-knowledge [8] if no other information but the truth of the statement being proved is revealed. Unfortunately, our protocol falls short of this goal because data privacy could *possibly* (not surely) be compromised by a malicious data consumer (see Section 5.4).

8. CONCLUSION

Crowd-sourced sensing has a bright future, but both the integrity of the collected data, and the privacy of data sources are always at risk. Without integrity assurances, data consumers like the government or researchers will be reluctant to use the data, and without privacy assurances, people will be reluctant to contribute the data.

We have proposed a possible solution using interactive proofs that *simultaneously* addresses the conflicting problems of integrity and privacy. The interactive proof allows an intermediary to convince a data consumer that it is accurately performing a privacy-preserving transformation with inputs from trusted sources, without providing those inputs to the consumer. Sources can be trusted when, for example, they provide verifiable attestations to the integrity of sensed data with the aid of integrated trusted platform modules [5]. The key idea is that unlike traditional interactive proofs with one prover (privacy proxy) and one verifier (data consumer), ours involves a collaboration between the verifier and an additional trusted party (data source) to keep the prover in check. We have provided soundness and correctness proofs for the protocol, discussed its limitations, and described its parameters and their effect on data integrity and privacy.

9. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under grants CISE-0747442 and CNS-1017034. We would like to thank Tom Shrimpton for his valuable feedback on the contents of this paper.

10. REFERENCES

[1] E. Agapie, E. Howard, J. Ryder, A. Steiner, D. Lam, R. Rosario, A. Modschein, D. Houston, J. Burke, M. Hansen, et al. PEIR. 2007.

[2] B. Barak. Lecture 15: Zero Knowledge Proofs. `www.cs.princeton.edu/courses/archive/spring10/cos433/lec17new.pdf`, Nov 2007.

[3] D. Chaum and E. Van Heyst. Group Signatures. *Berlin: Springer-Verlag*, 265, 1991.

[4] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security*, pages 21–21. USENIX Association, Berkeley, CA, USA, 2004.

[5] A. Dua, N. Bulusu, W. Feng, and W. Hu. Towards Trustworthy Participatory Sensing. In *HotSec'09: Proceedings of the 4th USENIX Workshop on Hot Topics in Security*. USENIX Association Berkeley, CA, USA, 2009.

[6] R. Ganti, N. Pham, Y. Tsai, and T. Abdelzaher. PoolView: stream privacy for grassroots participatory sensing. In *Proceedings of ACM SenSys*, pages 281–294, Raleigh, North Carolina, 2008. ACM.

[7] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178. ACM, 2009.

[8] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):728, 1991.

[9] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, page 304. ACM, 1985.

[10] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of ACM SenSys*, pages 125–138, Boulder, Colorado, 2006. ACM.

[11] A. Kapadia, N. Triandopoulos, C. Cornelius, D. Peebles, and D. Kotz. AnonySense: Opportunistic and Privacy Preserving Context Collection. *LNCS*, 5013:280, 2008.

[12] R. Popa, H. Balakrishnan, and A. Blumberg. VPriv: Protecting privacy in location-based vehicular services. In *Proceedings of the 18th Usenix Security Symposium*, 2009.

[13] S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, and M. Hansen. Image browsing, processing, and clustering for participatory sensing: lessons from a DietSense prototype. In *ACM SenSys*, pages 13–17, Cork, Ireland, 2007. ACM.

[14] J. Shi, R. Zhang, Y. Liu, and Y. Zhang. PriSense: Privacy-Preserving Data Aggregation in People-Centric Urban Sensing Systems. In *IEEE INFOCOM*, 2010.

[15] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty Fuzziness and Knowledge Based Systems*, 10(5):557–570, 2002.