Privacy-preserving Online Mixing of High Integrity Mobile Multi-user Data

Akshay Dua, Nirupama Bulusu, and Wu-chang Feng

Portland State University
{akshay,nbulusu,wuchang}@cs.pdx.edu

Abstract. Crowd-sourced sensing systems facilitate unprecedented insight into our local environments by leveraging voluntarily contributed data from the impressive array of smartphone sensors (GPS, audio, image, accelerometer, etc.). However, user participation in crowd-sourced sensing will be inhibited if people cannot trust the system to maintain their privacy. On the other hand, data modified for privacy may be of limited use to the system without mechanisms to verify integrity. In this paper, we present an interactive proof protocol that allows an intermediary to convince a data consumer that it is accurately performing a privacy-preserving transformation mixing inputs from multiple expected sources, but without revealing those inputs. Additionally, we discuss privacy transformation functions that are compatible with the protocol, and show that the protocol introduces very little overhead, making it ideal for real-time crowd-sourced data collection.

1 Introduction

Crowd-sourced sensing systems must protect the privacy of all data sources, whilst also providing integrity guarantees for the collected data. Data obtained from the crowd can enable novel people-centric applications in health care, traffic, and environmental monitoring systems [17, 15, 13]. User contribution of sensitive data such as location may be inhibited due to privacy concerns. One way to protect user privacy is to perform a privacy-preserving transformation, such as mixing, on the raw data collected from mobile users. But this may engender reluctance to trust the integrity of the transformation in the consumer. The conundrum here is that it is difficult to prove the transformation's integrity without revealing the raw data and compromising the privacy of data sources. Thus, integrity and privacy wind up as dueling goals of a crowd-sourced sensing system.

Most prior approaches have either proposed novel transformation functions to provide privacy [8, 16], or proposed mechanisms to verify data integrity [7], but have not addressed both problems simultaneously. VPriv [14] is the only prior work that attempts to offer both integrity and privacy using an additive homomorphic commitment scheme, for the application scenario of computing tolls over paths taken by vehicles. But its integrity is limited to additive functions, while its privacy is limited by the need for random spot checks.

Our previous work [6] provided a mechanism to verify the integrity of privacy-preserving transformations of data from an individual source without revealing the raw data. But it did not address the problem of verifying the integrity of privacy-preserving transformations that *mix* data from *multiple* data sources. Mixing data from multiple users, as opposed to performing transformations on data from a single user, is essential to ensuring better privacy for all users[8]. However, simultaneously achieving privacy and integrity with mixing is not trivial. It is reasonable to assume a user's vested interest in her own privacy, but not necessarily in the privacy of others. Thus, the integrity verification must now be robust to any privacy threats that stem from a collusion between a data source and the data consumer.

In this paper, we address the problem of privacy-preserving online mixing of mobile multi-user data. Our work uses the system model illustrated in Fig. 1. It assumes that multiple independent *data sources or producers* forward their raw data to a trusted *privacy proxy*. The proxy then performs a *privacy-preserving transformation* on the received data and forwards the result to a *data consumer*. Here, the proxy is analogous to a Tor mix-node [5], which mixes data from multiple sources to provide anonymity. Note that the proxy is trusted by the sources but not by the consumer.

Our goal is to enable the privacy proxy to assure the data consumer that, the result it publishes is indeed the output of a given privacy-preserving transformation on data from multiple expected sources as input (integrity guarantee), without having to provide that input to the consumer (privacy guarantee). The contributions of this paper include:

- An interactive proof protocol [10], using which, only an honest privacy proxy can convince a data consumer that it is correctly computing a given privacy-preserving transformation that mixes data from multiple users. Most importantly, the proof requires the proxy to send the consumer only the output of the transformation. Since the inputs — sensitive data contributed by participants — are never sent to the consumer, each participant's privacy remains protected (Section 4).
- Demonstrating privacy-preserving transformations whose integrity can be proved using our interactive proof protocol (Section 5).
- Deriving the overhead introduced by the protocol. We show that the overhead is a fraction of the complexity of the privacy-preserving transformation being computed by the proxy (Section 6)

2 Problem Statement

Only an honest privacy proxy P should be able to convince a data consumer C that it is indeed publishing the result of a privacy-preserving transformation function f_{priv} on data $D_j = \{d_{1j}, d_{2j}, ..., d_{nj}\}$ received from a set of sources $S = \{s_1, s_2, ..., s_n\}$ in interval j (Fig. 1). C receives the result $p_j = f_{priv}(D_j)$, but never the data D_j. Essentially, the following must be satisfied:

– **Integrity requirement.** C must be convinced with high probability that $p_j = f_{priv}(D_j)$ only if P is honest
– **Privacy requirement.** C must not be able to learn or verify that $D_j = \{d_{1j}, d_{2j}, ..., d_{nj}\}$

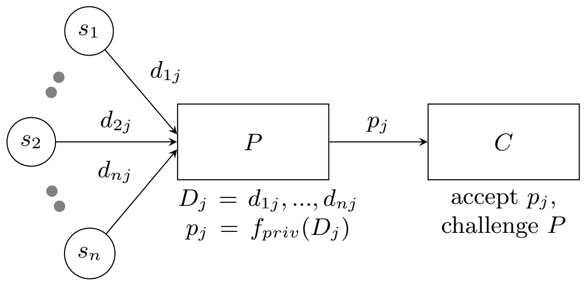

Fig. 1. System model

3 Threat Model

The design of our protocol aims to both prevent a malicious data consumer C from discovering any raw data D contributed by any of the sources S, and prevent a malicious privacy proxy P from using either alternate transformation functions besides f_{priv}, or data besides D, or both.

Insiders pose significant risks to the system. We assume that the data consumer is an adversary of privacy, but not of integrity. This is reasonable because the data consumer C has a vested interest in the integrity of the privacy-preserving transformation. Further, we assume that the privacy proxy P is an adversary of integrity, but not of privacy. The reason being, that the data sources trust P to protect their privacy, but the consumer may not trust P to preserve data integrity. An implication of the above adversarial model is that P and C do not collude in any way. Malicious data sources in S, on the other hand, could collude with the consumer C to compromise the privacy of other honest sources. This is reasonable because a source may have a vested interest in safeguarding her own privacy, but not necessarily in others' privacy. This is the most significant addition to the threat model discussed in [6]. Additionally, since sources could contribute bogus information to skew the collected data, they are considered adversaries of integrity. Our proof protocol alone, however, is not designed to address the threat of fabricated data from the sources. The proof only guarantees that the output of a privacy transformation was computed using inputs from S, but the integrity of those inputs — the sensory data collected by S – cannot be guaranteed by our protocol. Our earlier work [7] describes the design and implementation of a trusted sensing platform that can be used to address this issue.

s_i	**P**
interval j: sense d_{ij}	
if j mod $h = 0$,	
choose random m from $[0, h-1]$	
set $m = m + j$	
if $j = m$,	
save $[d_{ij}, j]$	
$s_i \rightarrow P$: d_{ij}	$P \rightarrow C$: $p_j = f_{priv}(d_{ij})$
...	...

Table 1. Normal Operation

We do not consider denial-of-service attacks in which communicating parties P, C, and S can potentially suppress responses that are expected by others. Nor do we consider threats from eavesdropping adversaries that can be mitigated by standard network security protocols, like TLS.

Our work focuses on the integrity and privacy of *content* rather than their *origin*. Thus, attacks that could reveal or alter the origin of a message are not considered. Directing traffic from S through a mix network like Tor [5] and using anonymous group signatures [3] for authentication can mitigate such attacks.

Finally, we assume that C honestly executes the protocol since it has a vested interest in collecting high-integrity data. However, C is free to perform offline privacy attacks on the data received from P. Mitigation strategies for such attacks have been addressed in the security analysis of our prior work [6], and are not explored in this paper.

4 Interactive Proof Protocol

As shown in Fig. 1, after receiving p_j, C may randomly choose to issue a challenge that will require P to prove that it is honestly computing f_{priv} using inputs from sources S. This challenge message marks the beginning of the interactive proof protocol. After the proof, C will be convinced about the integrity of the data with *high probability* only when P is honest, but not otherwise.

4.1 Preliminaries

For the protocol to work, we require a shared symmetric key k_{ic} between the source s_i and the data consumer C, a key k_{ip} between s_i and privacy proxy P, and a buffer at s_i that is large enough to store data collected in b distinct intervals. Where encryption/decryption is necessary, the notation $Enc_{k_{ic}}$ and $Dec_{k_{ic}}$ indicate a symmetric encryption and decryption algorithm (e.g. AES) using key k_{ic}. Additionally, we need the privacy-preserving transformation function f_{priv} to satisfy the following condition: given an *obfuscation function* $g(r, x)$ that obfuscates input x using random number r that we call the *obfuscation key*, we require that

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

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

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

Here, the privacy-preserving transformation is simply a mean of its inputs (raw data from sources) and we can see that this particular transformation satisfies Equation (1). What is significant, is that Equation (1) establishes a relationship between the *transformed value*, and *obfuscated raw data* from sources. This relationship is fundamental to the success of our protocol because it gives us the ability to check the integrity of the published data without requiring the potentially sensitive raw data from sources.

Data sources must, however, provide obfuscated data to the consumer during the protocol. Fortunately, without the obfuscation key, the consumer may have no way to extract the correct raw data, especially because the key changes every interval. Further, even if there was a way, say by using an oracle, the consumer could only ever extract raw data for half the number of intervals in which it challenges the proxy. So, if C challenges 4 out of 10 intervals, then using the oracle it could extract raw data for 2 intervals.

4.2 Protocol Details

Table 1 shows the normal operation of source s_i, which, continuously picks a random interval j from every h, and saves the corresponding data in its buffer. Once the buffer is full, s_i is ready to participate in the interactive proof protocol. We explain later how each source s_i picks the same j.

The interactive proof protocol begins once C randomly issues one of two challenges to P (via S, see Tab. 2). On receiving a response, C performs one of two different tests to check the response's integrity.

Both tests are performed using obfuscated raw data from sources in S or from their chosen leader s_{lead} (more on this later). Note that P *does not know* which test will be performed until it has responded. Thus, P's initial response acts as a bit commitment [2]. The tests serve three purposes:

1. Allow an honest P to pass either test with the same response, but force a dishonest P to create a different response for each. Since a dishonest P does not know which test is going to be performed, its chances of passing are 1/2. If P repeatedly passes, then C has more confidence in P's honesty.
2. Make sure that obfuscated raw data from S and the corresponding obfuscation key are not simultaneously available to C during any given challenge. If that were the case, C could extract the raw sensitive data from the obfuscated values. We can see, that during `Challenge 1`, C has the obfuscation key but not the obfuscated data. Where as in `Challenge 2`, its vice versa.

Sources $S = \{s_1, ..., s_n\}$	**C**
	for each $s_i \in S$,
	With probability 1/2:
	$C \rightarrow s_i$: $Enc_{k_{ic}}$(`Challenge 1`)
	OR, $C \rightarrow s_i$: $Enc_{k_{ic}}$(`Challenge 2`)
at each $s_i \in S$,	
decrypt challenge	
randomly choose saved interval l, $l \leq j$	
Sources $S = \{s_1, ..., s_n\}$	**P**
at each $s_i \in S$,	
if `Challenge 1`,	
Pick random number r	
$s_i \rightarrow P$: $M_{i0} = Enc_{k_{ip}}(g(r, d_{il}))$	
else,	
Pick random number r_i	
$s_i \rightarrow P$: $M_{i0} = Enc_{k_{ip}}(g(r_i, d_{il}))$	
	$O_l = Dec_{k_{1p}}(M_{10}), ..., Dec_{k_{np}}(M_{n0})$
	$P \rightarrow C$: $p = f_{priv}(O_l)$
Sources $S = \{s_1, ..., s_n\}$	**C**
if `Challenge 1`,	
$s_{lead} \rightarrow C$: $M_1 = Enc_{k_{lead,c}}(r, l)$	
else,	
at each $s_i \in S$,	
$s_i \rightarrow C$: $M_{i2} = Enc_{k_{ic}}(g(r_i, d_{il}))$	
	M_1 OR $M_{12}, ..., M_{n2}$
	Test 1 (if `Challenge 1`):
	$r, l = Dec_{k_{lead,c}}(M_1)$
	if $p \neq g(r, p_l)$,
	reject
	Test 2 (if `Challenge 2`):
	$O_l = Dec_{k_{1c}}(M_{12}), ..., Dec_{k_{nc}}(M_{n2})$
	if $p \neq f_{priv}(O_l)$,
	reject

Table 2. Interactive proof protocol

3. Use one test to check that a published transformed value was indeed computed using raw data from s_i, and the other to check that the privacy-preserving transformation computed by P was indeed f_{priv}. Test 1 does the former while Test 2 does the latter. Test 1 compares $f_{priv}(O_l) \stackrel{?}{=} g(r, f_{priv}(D_l))$, where O_l is obfuscated sensory data, and D_l is raw sensory data for a past interval l. This will be true only if source s_i created O_l by obfuscating D_l. Test 2 compares the transformed value sent by P with $f_{priv}(O_l)$ computed by C. Since C is computing f_{priv} on data received directly from the sources, this comparison will be true only if P computed f_{priv} as well. The reason for using a different obfuscation key (r_i) per source in `Challenge 2` is to prevent a collusion attack where a malicious source

could have leaked the common key to C and compromised the privacy of other honest sources.

Once the interactive proof is complete, s_i purges the respective interval of data from its buffer, thus making room for more. In the interest of clarity, we have omitted the use of digital signatures for authentication.

One question remains: how does one get all the sources to pick the same saved interval j, same challenge interval l, and random number r? With the correct r, j, and l, each source can obfuscate and forward its own data to C as shown in Tab. 2.

```
**Sources** $S = \{s_1, ..., s_n\}$
establish shared group key $k_g$
establish leader $s_{lead} \in S$
$s_{lead}$ broadcasts $M_{init} = Enc_{k_g}(r_{seed}, b, h)$
**at each** $s_i \in S$,
  set $r_{seed}, b, h = Dec_{k_g}(M_{init})$
  initialize PRNG with $r_{seed}$
  allocate buffer for $b$ intervals of data
```

Fig. 2. Initialization phase.

We can get all the sources to pick the same random numbers if each one uses the same Pseudo-Random Number Generator (PRNG) with the same random seeds. The random seed could be securely communicated (via P) to each source by another that is picked to be the leader (s_{lead}). For secure communication among sources, a group key will need to be established [12]. To reliably elect a leader, a communication-efficient stable leader election protocol can be used [1].

Once the secure communication channels are established, the elected leader can broadcast (via P) the protocol's various parameters — the random seed, size of the buffer b, and sampling frame size h — to the rest of the sources during an initialization phase (Table 2).

5 Privacy-Preserving Transformations

We do not attempt to define what precisely is a privacy-preserving transformation. Rather, we claim that if such a transformation satisfies Equation (1), then the entity computing the transformation can provide proofs of integrity for the result without disclosing the inputs. The question is, what privacy-preserving transformations satisfy Equation (1)?

While providing an exhaustive list (or category) of privacy-preserving transformations satisfying Equation (1) is out of the scope of this paper, we present one whose computational integrity can be proven using our protocol. The privacy-preserving transformation in question is a *cloaking* algorithm similar to the one by Gruteser and Grunwald [11] intended for use in Location Based Services (LBS). The idea is to spatially cloak a set of GPS coordinates published by participants, and return an area (e.g., a quadrant) that includes at

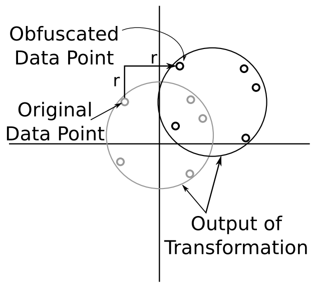

Fig. 3. Spatial Cloaking

least k_{min} of them. Here, k_{min} quantifies the amount of anonymity desired. Specifically, it enforces the fact that each participant's location cannot be distinguished from at least $k_{min} - 1$ others.

The privacy-preserving transformation f_{priv} is the function that takes the GPS coordinates as input and returns the respective area (see Fig. 3) as output. Unlike the original algorithm, where the f_{priv} returned a quadrant, we define an f_{priv} that returns a circular area (x, y, u), where (x, y) is the center of that circle, and u is its radius. Formally, we define f_{priv} as:

$$f_{priv}((x_1, y_1, 0), ..., (x_n, y_n, 0)) = (x_c, y_c, u_c) \tag{2}$$

Intuitively, the privacy transformation above returns a circle large enough to cover all the circles provided as input. A participant's coordinates are thus expressed not as points, but as circles whose radii are zero. We now define the obfuscation function g as: $g(r, (x, y, u)) = (x + r, y + r, u)$.

Intuitively again, the obfuscation function moves the input circle by r units in the x and y dimensions. Using the above definitions for f_{priv} and g in the left and right hand side of Equation (1) we have:

$$\begin{array}{ll} f_{priv}(g(r,(x_1,y_1,0)),...,g(r,(x_n,y_n,0))) & g(r, f_{priv}((x_1,y_1,0),...,(x_n,y_n,0))) \\ =f_{priv}((x_1+r,y_1+r,0),...,(x_n+r,y_n+r,0)) & =g(r,(x_c,y_c,u_c)) \\ =(x_c+r,y_c+r,u_c) & =(x_c+r,y_c+r,u_c) \end{array}$$

Since f_{priv} and g satisfy Equation (1), our interactive proof protocol is applicable in this LBS scenario.

6 Overhead

We now present an analysis of the overhead imposed by our protocol. We will show that the introduced overhead is a fraction of the complexity of the privacy-preserving transformation computed by the proxy P. Our baseline is a data collection system like PoolView [8], where privacy guarantees are provided without integrity guarantees. The overhead then, is due to all computations and message exchanges required to perform the interactive proof of integrity while preserving privacy (Table 2). Also, note that we are mainly interested in the overhead of the more expensive challenge where in addition to P, C must also compute f_{priv}. Our result, therefore, is the worst case bound for overhead as it assumes that every challenge from C is the more expensive one, when in reality that will only be true during approximately half the challenges.

We define t_b as the time it takes for baseline operation: sources in S send data to P, which computes f_{priv} over that data and forwards the result to C. We define t_c as $t_b + t_{proof}$ where t_{proof} is the time it takes to complete the interactive proof. Then, overhead $t_o = t_c - t_b = t_{proof}$. Now,

$$t_b = t_{fpriv} + \frac{rtt_{SP}}{2} + \frac{rtt_{PC}}{2}$$

where, t_{fpriv} is the time it takes to compute the privacy transformation, and rtt_{ij} is the round-trip time between i and j. We have excluded minor computations such as a data source randomly choosing and saving an interval of data in its buffer. Also,

$$t_c = 3\ t_{fpriv} + 2\ rtt_{SP} + 2\ rtt_{PC} + \delta$$

where δ includes symmetric encryption or decryption operations performed on raw data values, and the computation of the simple obfuscation function $g(r, x)$.

Subtracting t_c from t_b we get overhead:

$$t_o = 2\ t_{fpriv} + \frac{3\ rtt_{SP}}{2} + \frac{3\ rtt_{PC}}{2} + \delta$$

Note that the overhead t_o is applicable only in those intervals in which the data consumer issues a challenge. The probability $P(challenge)$ with which the consumer issues a challenge during any given interval is $1/h$ (Section). Further, if the round-trip, encryption, and decryption times are negligible compared to the privacy transformation f_{priv}, then $t_o \approx 2\ t_{fpriv}$. With this in mind, the expected overhead of our protocol after I intervals of data collection is,

$$E(overhead) = I \times t_o \times P(challenge) = \frac{2I\ t_{fpriv}}{h}$$

It is important to note that if the data publishing interval is larger than $E(overhead)$, then the entire proof will finish before the sources disseminate the next interval of data. Thus, causing no perceptible delay in data publication.

7 Conclusion

Crowd-sourced sensing can revolutionize applications from intelligent transportation to health care monitoring, but confronts the challenge of maintaining user privacy to encourage contribution of data, while maintaining data integrity to encourage governmental and citizen use of that data. We have proposed the first solution using interactive proofs that allows an intermediary to convince a data consumer that it is accurately performing a privacy-preserving transformation that mixes inputs from multiple data sources, without providing those inputs to the consumer. The proposed protocol preserves the privacy advantages of mixing data from multiple sources, while being robust to privacy threats that arise from collusion between a data source and a consumer during integrity verification. 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 all additional parties that wants to protect their privacy (data sources) to keep the prover in check. We have analyzed the protocol overhead and discussed compatible privacy-preserving transformations.

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 feedback on the interactive proof protocol.

References

1. M. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg. Stable leader election. *Distributed Computing*, pages 108–122, 2001.
2. D. Chaum, I. Damgård, and J. van de Graaf. Multiparty computations ensuring privacy of each partys input and correctness of the result. In *Advances in Cryptology*, pages 87–119. Springer, 1987.
3. D. Chaum and E. Van Heyst. Group Signatures. *Berlin: Springer-Verlag*, 265, 1991.
4. S. Consolvo, D. McDonald, T. Toscos, M. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, et al. Activity sensing in the wild: a field trial of ubifit garden. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1797–1806. ACM, 2008.
5. 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.
6. A. Dua, N. Bulusu, and W. Feng. Catching Cheats with Interactive Proofs: Privacy-preserving Crowd-sourced Data Collection Without Compromising Integrity. 2010.
7. 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.
8. 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.
9. 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.
10. 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.
11. M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.
12. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. *Journal of Cryptology*, 20(1):85–113, 2007.
13. E. Paulos, R. Honicky, and E. Goodman. Sensing atmosphere. In *Workshop on Sensing on Everyday Mobile Phones in Support of Participatory Research*. Citeseer, 2007.
14. R. Popa, H. Balakrishnan, and A. Blumberg. VPriv: Protecting privacy in location-based vehicular services. In *Proceedings of the 18th Usenix Security Symposium*, 2009.
15. 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.
16. J. Shi, R. Zhang, Y. Liu, and Y. Zhang. PriSense: Privacy-Preserving Data Aggregation in People-Centric Urban Sensing Systems. In *IEEE INFOCOM*, 2010.
17. A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. VTrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98. ACM, 2009.