Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Privacy-preserving computation of participatory noise maps in the cloud☆

George Drosatos[a,*], Pavlos S. Efraimidis[a], Ioannis N. Athanasiadis[a], Matthias Stevens[b,c], Ellie D'Hondt[c]

[a] Department of Electrical & Computer Engineering, Democritus University of Thrace, GR671 00 Xanthi, Greece
[b] ExCiteS Group, Department of Civil, Environmental and Geomatic Engineering, University College London, London, United Kingdom
[c] BrusSense Team, Department of Computer Science, Vrije Universiteit Brussel, Brussels, Belgium

ARTICLE INFO

Article history:
Received 28 September 2012
Received in revised form
15 November 2013
Accepted 23 January 2014
Available online 10 February 2014

Keywords:
Privacy-preserving computation
Cloud computing
Participatory sensing

ABSTRACT

This paper presents a privacy-preserving system for participatory sensing, which relies on cryptographic techniques and distributed computations in the cloud. Each individual user is represented by a personal software agent, deployed in the cloud, where it collaborates on distributed computations without loss of privacy, including with respect to the cloud service providers. We present a generic system architecture involving a cryptographic protocol based on a homomorphic encryption scheme for aggregating sensing data into maps, and demonstrate security in the Honest-But-Curious model both for the users and the cloud service providers. We validate our system in the context of NoiseTube, a participatory sensing framework for noise pollution, presenting experiments with real and artificially generated data sets, and a demo on a heterogeneous set of commercial cloud providers. To the best of our knowledge our system is the first operational privacy-preserving system for participatory sensing. While our validation pertains to the noise domain, the approach used is applicable in any crowd-sourcing application relying on location-based contributions of citizens where maps are produced by aggregating data – also beyond the domain of environmental monitoring.

1. Introduction

Many people are reluctant to entrust today's computer systems with their personal information, thus Mundie et al. (2002) have identified privacy as a pillar of trustworthy software systems. Specifically, the authors consider trustworthy systems to respect *privacy* if *the customer is able to control data about themselves, and those using such data adhere to fair information principles*. This work contributes to this end, by presenting an architecture and implementation for incorporating privacy-preserving techniques in participatory sensing applications.

Participatory sensing (Burke et al., 2006; Paulos, 2009) appropriates everyday devices such as mobile phones to acquire information about the physical world (and the people in it) at a level of granularity which is very hard to achieve otherwise. A crucial component of participatory sensing systems is *geolocation*, i.e. labelling data with geographic coordinates. For example, in the context of *NoiseTube* (Maisonneuve et al., 2010; Stevens, 2012), a participatory sensing system and service[1] designed to monitor and map noise pollution, it would be practically impossible to produce noise maps on the basis of sound level measurements, gathered quasi-continuously as contributors walk the streets, without automatic geolocation of measurements by means of GPS (Global Positioning System). As a recent survey shows (Christin et al., 2011), the same situation applies more generally, as the potential of fine-grained measurements essential to participatory sensing frameworks is only manageable if this data can be automatically organized through location.

However, location traces constitute sensitive personal information. In small-scale deployments, in which individual contributors know or trust each other, the disclosure of such information may be acceptable. However, in larger-scale deployments, involving more contributors and possibly coordinated by some authority, trust relationships tend to be much weaker and contributors may be uncomfortable about the type of information that is collected, and with whom it is shared. Hence, scaling up a participatory sensing

☆ A preliminary version of this work has been published in 36th Annual IEEE Computer Software and Applications Conference (COMPSAC 2012).

* Corresponding author. Tel.: +30 25410 79756; fax: +30 25410 79756.
E-mail addresses: gdrosato@ee.duth.gr (G. Drosatos), pefraimi@ee.duth.gr (P.S. Efraimidis), ioannis@athanasiadis.info (I.N. Athanasiadis), m.stevens@ucl.ac.uk (M. Stevens), eldhondt@vub.ac.be (E. D'Hondt).

[1] http://www.noisetube.net.

0164-1212/$ – see front matter © 2014 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2014.01.035

project inherently increases privacy concerns (Lane et al., 2010; Christin et al., 2011), which in turn can severely hamper the project from reaching its goals.

In this paper, we present a privacy-preserving solution for participatory sensing frameworks where location-based data aggregation is used to produce maps involving measurements contributed by groups of users. Our system, called *NoiseTubePrime* or *NTPrime* for short, relies on privacy-preserving distributed computation in the cloud and is oriented towards noise mapping campaigns set up by citizens, researchers or authorities. NoiseTubePrime differs from earlier work on privacy-preserving mobile sensing systems, i.e., (Kapadia et al., 2008; Becchetti et al., 2010; Shi et al., 2010), as it is at the same time secure, correct, and transparent to end-users. The core of the NoiseTubePrime architecture is a privacy-preserving cryptographic protocol implementing a large scale distributed computation [see for example, Bilogrevic et al., 2011; Drosatos and Efraimidis, 2014].

The novelty of our approach is first, that by thinking in terms of campaigns rather than when thinking in terms of privacy of stand-alone users one can deal with privacy in a distributed way. This allows us to avoid the trade-off between user privacy and the accuracy of the resulting maps (as e.g. data obfuscation does). Campaigns are collective sensing efforts by groups of users, where geographical, temporal and/or contextual constraints determine the measurements under consideration. The outcome of a campaign entails aggregating individual user data into a composite map, and it is precisely this property that allows us to rely on a distributed cryptographic protocol which ensures privacy of users and at the same time precise noise maps. Second, our approach is the first to incorporate cloud computing, essential to ensure transparency and efficiency. As we discuss below, the main reasons for resorting to computation in the cloud are high availability, scalability, and ease of deployment. Moreover, the way we use the cloud simplifies the privacy-preserving protocol that assures user privacy.

We support the above claims in the remainder of this paper, which is organized as follows. We present key concepts in Section 2, introducing participatory noise monitoring, personal data and privacy, cloud computing and agent-based computation. Section 3 presents the NoiseTubePrime system architecture and its most pertinent use cases, while Section 4 details the privacy-preserving computation protocol that NoiseTubePrime implements. Next, Section 5 focuses on the actual deployment of NoiseTubePrime in the cloud as well as its validation in terms of concrete noise mapping experiments, while Section 6 identifies key innovations of our approach with respect to previous work. The paper concludes with Section 7, where we also propose directions for future research.

2. Background

2.1. Participatory monitoring campaigns

Participatory sensing platforms, are typically client-server systems that consist of a mobile application used by contributors on the one hand, and a central server application[2] on the other. The former enables users to sense environmental parameters (e.g. sound level) whenever and wherever they please. The latter serves to receive and store measurement data which users uploaded to it (either automatically or manually) and to generate outputs such as visualizations of various kinds. A typical visualization is a map that shows the geographical distribution of the measured parameter, e.g. noise maps in the case of NoiseTube (Maisonneuve et al., 2010; Stevens, 2012). Such maps can be based on geolocated data contributed by a single or multiple users. Either way, users are effectively sharing personal location traces with a system, and other users, which they may or may not trust.

NoiseTube is one of the first participatory sensing platforms to endeavour the transition from a tool used by individuals to one that can serve as a basis for coordinated measurement campaigns, be it grassroots or authority-led. Indeed, recent work (D'Hondt and Stevens, 2011; D'Hondt et al., 2013; Stevens, 2012) shows that, when coordinated properly, NoiseTube campaigns can produce collective noise maps that are of comparable quality to simulation-based maps produced by governments today. To do this a statistical component was introduced which produces a single aggregate noise map from a collection of measurement tracks contributed by groups of users. The basic procedure is this: divide the surveyed area into smaller areas using a regular grid, partition the set of measurements over those areas based on their geographic coordinates, make a statistical analysis per unit area, and finally, map the colour coded averages on each pertaining area. While such aggregated map-making has been carried out before in individual instances, we are currently extending the NoiseTube Web application with this collective noise mapping functionality so that it enables community-driven environmental sensing. The idea is that larger and more diverse groups of people may use NoiseTube to define and coordinate their own campaigns with little to no involvement of experts. Concrete examples are citizens that wish to map the noise in their commune or city while construction works are going on, a commune which wants to investigate how rescheduling of a bus line effects Monday morning traffic, or a researcher who wishes to compare peak hours in various cities.

In small-scale campaigns, such as the one reported in D'Hondt et al. (2013), privacy issues tend to be of little concern. The reason is that participants typically already know and trust each other (e.g. because they are members of a citizen activist group), consciously take part in a scientific experiment or community effort and got time to get acquainted with the researchers and/or coordinators in person. However, in campaigns that cover larger areas, last longer, and involve larger numbers of more diverse contributors and coordinators, this kind of mutual confidence could easily break down.

The issue of privacy is thus an important hurdle for the adoption of tools such as NoiseTube for larger-scale (e.g. city-wide) mapping campaigns, a situation which holds more generally. Hence there is a clear need for a privacy-preserving extension of participatory sensing platforms. In this work we design a privacy-preserving extension of participatory sensing frameworks, introducing privacy-preserving functionalities at several levels, and implement it in the context of the NoiseTube platform. As a proof of concept, as well as a validation of correctness, we use data from the above-mentioned earlier experiments to demonstrate that NoiseTubePrime produces exactly the same maps in a privacy-preserving way.

2.2. Personal data and privacy

Desktop, mobile computing and sensing technology have greatly increased the amount of personal information that is generated, while recent advances of database technology enable the potential for this information to be (permanently) stored and processed. Recent revelations about mass-surveillance of major Internet services by government intelligence agencies have once again started a worldwide debate about the precariousness of personal privacy in a world which is ever more reliant on connectivity and "big data" (Streitfeld and Hardy, 2013; Buytendijk and Heiser, 2013). Personal data is a critical, valuable resource that has to be protected in order to ensure the individual's privacy rights: to protect his/her privacy by retaining the control over his/her personal

[2] Sometimes called a *community memory* (Steels, 2007; Stevens, 2012).

data and knowing who, when and why gets access to this data (Efraimidis et al., 2009). At the same time, the wide acceptance of electronic transactions for everyday tasks resulted in an abundance of applications that rely on the processing of this personal data. Thus, locking all personal data away is not a solution. Instead, it should be possible to process such data in a way that is both efficient and ensures its protection. Furthermore, when an individual makes a transaction, only the minimum amount of personal information that is needed to complete it should be disclosed, with clear terms on how the personal data will be used.

The issue of privacy in mobile participatory sensing applications was recently surveyed in Christin et al. (2011). In this context, privacy is redefined as "the guarantee that participants maintain control over the release of their sensitive information, which includes [. . .] information that can be inferred from both the sensor readings themselves as well as from the interaction of the users with the participatory sensing system". Here sensitive information typically exists in the form of location traces (as in NoiseTube), but can also reside in sensing data such as sound samples, pictures, or health data. It is important to realize that even for anonymous users analysis of location traces may infer their location based on their residence location and reverse white pages lookups. As we shall see below, the tight decoupling of data gathering and processing that our system proposes, compared with the cryptographic algorithm used, ensures that these issues cannot arise. We come back to the related work discussed in this survey and beyond, and the differences with our approach, in Section 6.

Concretely, to preserve user privacy in this setting we follow an approach similar to one described in the Polis Project (Efraimidis et al., 2009). Here each user is represented by a personal software agent, who manages his/her personal data and controls access to this data. Third-party applications, other agents or services direct requests at these personal software agents rather than at the user himself. The agents respond to these requests according to a corresponding license agreement or policy. For the needs of this work, we adapted the Polis approach to the management of personal data of participatory sensing. However, the most important difference of this work with respect to previous applications of the Polis approach, is that in NoiseTubePrime, the personal software agents are outsourced to the cloud.

2.3. *Cloud computing*

The past few years have seen a shift towards the adoption of cloud computing technology, by industry and governments alike. Computing infrastructure, instead of being offered *as a product*, rather is offered *as a service*. Instead of running on machines owned or controlled by the user or client these services run "in the cloud".[3] In this way server hardware, storage resources, computational capacity, and software are designed, managed and delivered as cloud-based services.

An important novelty of the NoiseTubePrime approach is that personal software agents, loaded with personal data in encrypted form and guarding each user's privacy concerns, are outsourced to existing commercial cloud infrastructures. This relieves the user from the trouble to run and manage his/her own software agent. In NoiseTubePrime, personal agents are implemented as Web services, deployed in the cloud.

The main reasons for resorting to computation in the cloud in the context of privacy-preserving participatory sensing are high availability, scalability, ease of deployment and privacy. The need for high availability is essential because we need to ensure that all the data collected by different campaign contributors is available every time an aggregated map is to be generated. Concretely this means that a piece of software representing each contributor (i.e. an agent) must be *online* and able to respond to outside requests at all times. While in principle this is feasible with a smartphone application (cf. chat applications), mobile data connectivity can be intermittent and local computational resources are limited. Indeed, while noise maps do not need to be produced real-time, their computation is computationally intensive (D'Hondt et al., 2013), and the cryptographic extension only increases this issue. The fact that cloud computing services are extremely scalable, means that sensing campaigns can grow without the coordinators having to worry about things like server load and network bandwidth. Ease of deployment is an important concern because one cannot expect campaign contributors to install and configure complicated software on their personal computers, let alone run their own servers. Hence low-cost or even free cloud computing services that allow people with relatively moderate computing skills to set-up and manage their own software agent with a few clicks (using a deployment package provided to them) offer a suitable alternative. Finally having users run a "server-like" application on their personal phone could raise additional privacy concerns and could affect battery life. Choosing for a cloud service solution means that we decouple the role of collecting data on a mobile phone (using the NoiseTube Mobile app) from the role of managing the (protected) data and taking part in distributed computations (through the NoiseTubePrime software agent). When this chain of custody is respected the central NoiseTube service does not have access to private, sensitive information under no circumstances. For the above reasons it is preferable to host the software agents on an infrastructure with near-permanent availability and vast computational resources.

Today, there exist several commercial cloud computing providers which offer services at very affordable rates and in some cases (provided that the bandwidth and computational requirements are small) even for free. While cloud technology significantly changed how computing infrastructure is offered, there are two issues that need to be taken into account: one is interoperability, and the second is privacy. Indeed, cloud computing comes in different flavours from various vendors, and as standardized APIs are still largely lacking, deploying similar services in each one of them can require significant efforts. The other issue is that of privacy, as cloud technology has been criticized in terms of the potential for cloud service providers to gain access to personal data.

In what follows we will explain how our architecture deals with both privacy and interoperability issues, as it does not disclose any personal data to the cloud service providers (all data is encrypted), while we demonstrate it over a heterogeneous environment of different service providers.

3. The NoiseTubePrime system

In order to remedy privacy concerns when creating collective maps, we propose a solution relying on a privacy-preserving distributed computation algorithm for generating grid-based maps for a target area and time-frame. Here we are inspired by the procedure used in the existing NoiseTube service (D'Hondt et al., 2013; Stevens, 2012), though we stress that our ideas hold more generally for any participatory sensing framework where maps are produced in terms of aggregated measurements. However, for the sake of convenience we phrase our explanations below in terms of noise. At the basis of this algorithm lies a privacy-preserving cryptographic protocol for secure multi-party computations (Yao, 1982), which inputs are current or archived datasets of geolocated sound level measurements gathered by multiple users. Computation is executed by software agents running in the cloud.

[3] I.e. on servers in vast, remote data centres with high bandwidth and high reliability, operated and maintained by cloud service providers.

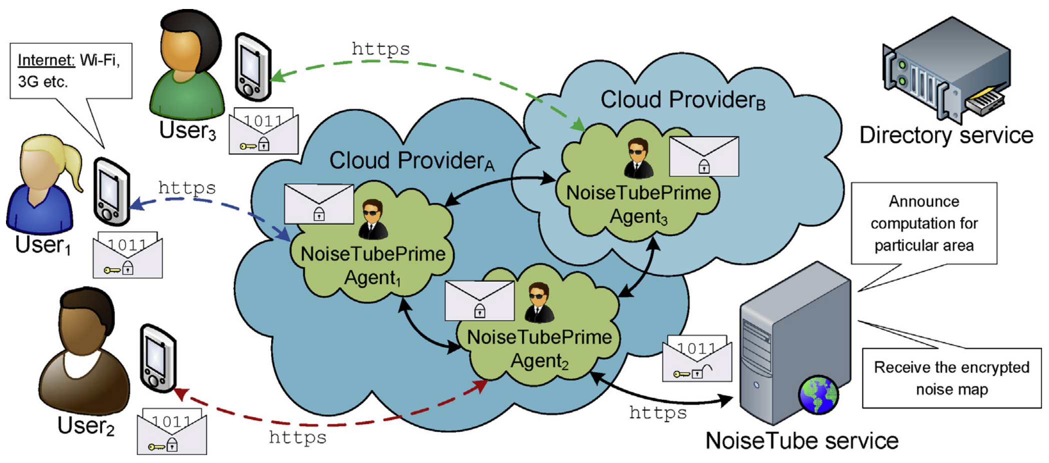


Fig. 1. An example of an aggregate noise map.

Each user is represented by a personal, cloud-based software agent which acts as a mediator. Such an agent temporarily stores encrypted user data, takes part in the generation of participatory maps on the user's behalf, while also crucially preserving his/her privacy. All data transmitted by users to NoiseTubePrime agents is encrypted. In this way we overcome privacy issues related to how the cloud service provider might treat the data. Cloud deployment ensures that agents are online continuously and have adequate computational resources. In this way users do not need to operate agents on their mobile phones or personal servers.

3.1. *The general architecture of our system*

An architectural diagram of the NoiseTubePrime system is shown in Fig. 1. A typical scenario proceeds as follows. Suppose a particular entity, be it an authority or a citizens' organization, is interested to map a local area during a time span of interest (e.g. Friday night in a pub area). The initiative taker(s) then organize a measurement campaign in which a group of citizens use the NoiseTube system to gather geolocated sound level measurements in the specific geographical region and time period. The campaign proceeds through the following steps (Fig. 2):

(a) To collect data about noise pollution users download the NoiseTube client application for their mobile device (e.g. from the Google Play app store).[4] By default measurement data is stored locally on each user's device. Users set up their personal cloud agent,[5] which registers to a *Directory Service* (DS) for the virtual network topology we deploy. Each user mandates his/her NoiseTubePrime agent to take part in existing or future campaigns, following a user-specified privacy policy.

(b) At some point in time the NoiseTube service announces a new campaign. Users are invited to participate through their agents, where agent policy dictates how agents should respond to such requests. For instance, agents may choose to participate to campaigns based on whether their owners plan to collect data in the specific region or not, or have collected relevant data before.[6] A deadline is set for all agents interested in contributing to register via the DS.

(c) When a user agrees to join a campaign (through the mediating agent), his/her mobile device inspects the user's local dataset for measurements that satisfy the given constraints, and uses this data to generate and encrypt the contribution of the user. The encrypted contribution is then handed over to the user's NoiseTubePrime agent in the cloud as soon as connectivity is available. This data upload operation takes place once per user and campaign/computation, and from that point on the user's mobile device is no longer involved in the computation.

(d) Each NoiseTubePrime agent manages a user's private data in the form of an encrypted map for the area of interest. Maps are encrypted with a public key that is either used across the system, is specific for the campaign in question, or for a specific time period. This public key is one of a public–private key pair that was generated by the NoiseTube service for this purpose; the private key is kept only by the NoiseTube service itself. Agents only use the encrypted data to participate in the generation of collective maps, when allowed to do so by user policy.

(e) After the announced deadline has passed the NoiseTube service initiates the distributed computation in the cloud. Note that agents from different users can be hosted on different cloud services. A list of the participating agents is retrieved from the DS. Agents are organized into a virtual network topology in which distributed computations take place. This may be a simple ring topology or something more sophisticated such as a tree for time-critical computations. One of the agents is selected to operate as the root-node for the specific computation via an appropriate request.

(f) The root-node coordinates a distributed computation that generates the specified noise map. This algorithm is detailed in Section 4.

(g) When agent interactions for the distributed computation are over, the NoiseTube service receives an encrypted aggregate noise map without any trace of the personal data of individual users. The NoiseTube service, using its private key, decrypts the received data to obtain the requested noise map, which is then made available accordingly. Interested parties can log on to the service to visualize and explore the resulting noise maps. A user's private information is not disclosed at any stage of the participatory noise mapping process.

4. 4 The privacy-preserving computation

Privacy-preserving computation is a large field which encompasses many challenges and tools. The theoretical foundations

[4] Note that the NoiseTubePrime functionality is not yet incorporated in publicly available version of the NoiseTube Mobile app.
[5] In the future, this activity could be automated through the mobile app or the NoiseTubePrime website.
[6] Hence the computations may involve both past, current or future data.

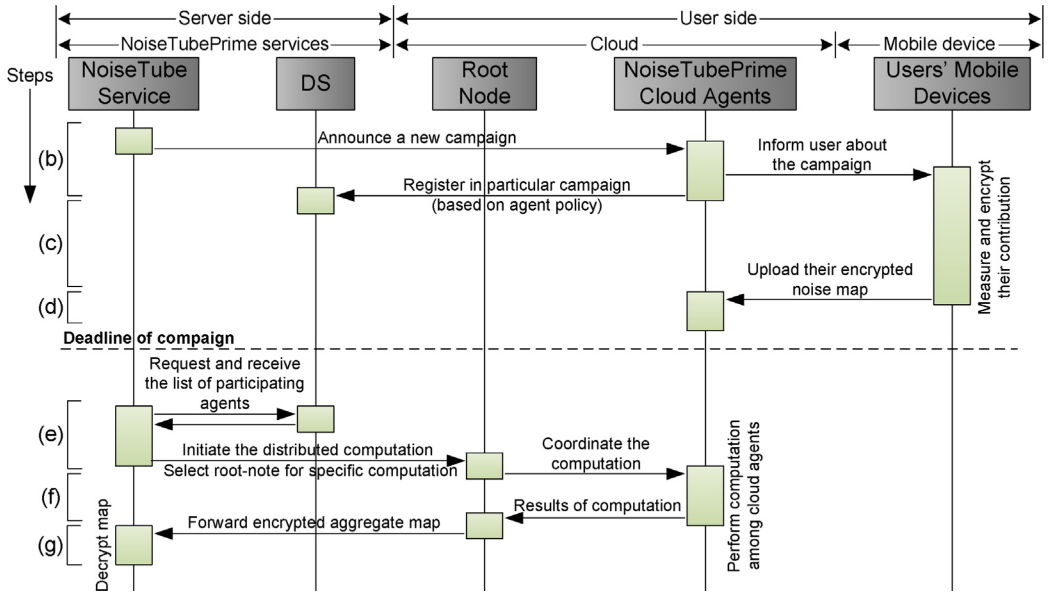

Server side
User side
NoiseTubePrime services
Cloud
Mobile device
Steps
NoiseTube Service
DS
Root Node
NoiseTubePrime Cloud Agents
Users' Mobile Devices
(b)
(c)
(d)
Announce a new campaign
Inform user about the campaign
Register in particular campaign (based on agent policy)
Measure and encrypt their contribution
Upload their encrypted noise map
Deadline of compaign
Request and receive the list of participating agents
(e)
(f)
(g)
Initiate the distributed computation
Select root-note for specific computation
Coordinate the computation
Results of computation
Forward encrypted aggregate map
Decrypt map
Perform computation among cloud agents

Fig. 2. Interaction diagram of the NoiseTubePrime system.

have been set since the seminal work of Yao on secure multi-party computations (Yao, 1982). It is known that theoretically any distributed computation can be converted to a secure multi-party computation (which can be used within a privacy-preserving computation). The problem is that the general theoretical solutions are computationally very demanding, to the extent that they are considered impractical for almost any practical application. The major challenge then, is to build efficient privacy-preserving solutions to real applications. Indeed, it is possible to derive optimized specialized solutions for particular problems. Such an example is the NoiseTubePrime application presented in this work. Moreover, NoiseTubePrime is also a new approach for outsourcing computations to the cloud in a privacy-preserving manner.

In the rest of this section, we describe the cryptographic protocol for calculating aggregated sensing maps in a privacy-preserving way, which is implemented by the NoiseTubePrime agents. The communication between agents in our protocol is performed over secure sockets (SSL/TLS). The protocol is secure in the *Honest-But-Curious* (HBC) model (see Definition 4 in Section 4.4). We also assume that the cloud providers are honest-but-curious, and that they do not collude with NoiseTube to reveal user data. We note that, if need be, the latter threat can be addressed by deploying a threshold decryption scheme (Damgård and Jurik, 2001).

4.1. The PrimeNoiseMap problem definition

The main goal of our work is to generate aggregate noise maps without violating the privacy of participants. The personal data which is needed for the computation are sensor (in our case sound level) measurements, associated with the user location and time-stamp, compatible with a particular campaign. To formalize the problem addressed in this work, we define the abstract *PrimeNoiseMap* problem for the privacy-preserving computation of participatory noise maps related to a particular measurement campaign. NoiseTubePrime is then an approach and associated system that solves the PrimeNoiseMap problem.

Definition 1 *(The PrimeNoiseMap Problem). An instance of the PrimeNoiseMap problem consists of:*

- **N users** $u_1, u_2, \ldots, u_N$ *and their geolocated, timestamped sound level measurements, where N is the number of participants that contribute to the campaign.*
- **Input:** *The geographic area of interest (defined by minimum and maximum latitudes and longitudes) together with the cell dimensions (e.g. 20 m × 20 m) of a grid covering that area, the time intervals of interest, the deadline for the distributed computation and a public encryption key.*
- **Output:** *The aggregated noise map with the required statistical information, i.e., number of noise measurements and their average value, per grid cell.*

We should note that the PrimeNoiseMap problem can be easily generalized to pertain to different kinds of sensor measurements (e.g. temperature instead of sound level) and is thus relevant to other participatory sensing systems and scenarios as well.

4.2. The distributed protocol

We present a protocol for a privacy-preserving computation that solves the PrimeNoiseMap problem. The protocol does not disclose any locations, timestamps or sound level measurements of any participants; only the final aggregate noise map is revealed at the end of the computation.

Initially, the NoiseTube service announces that a specific campaign is planned. The announcement includes the campaign name, the area and time period of interest, the public encryption key and the response deadline. When the campaign's deadline is reached each NoiseTubePrime agent, registered with the DS for that specific campaign, receives a request for the distributed computation as well as a corresponding deadline. Within the deadline, each agent communicates with the user's mobile device, and asks for any data that is relevant to the specific computation instance. The user is involved in the particular computation according to his/her privacy policy.[7]

[7] User privacy policy can be quite sophisticated: User may contribute to all campaigns, even if there is no data, or only to certain ones selected manually with care. While such a broad spectrum of strategies for user policy can be supported by our system, it is not the focus of this work.

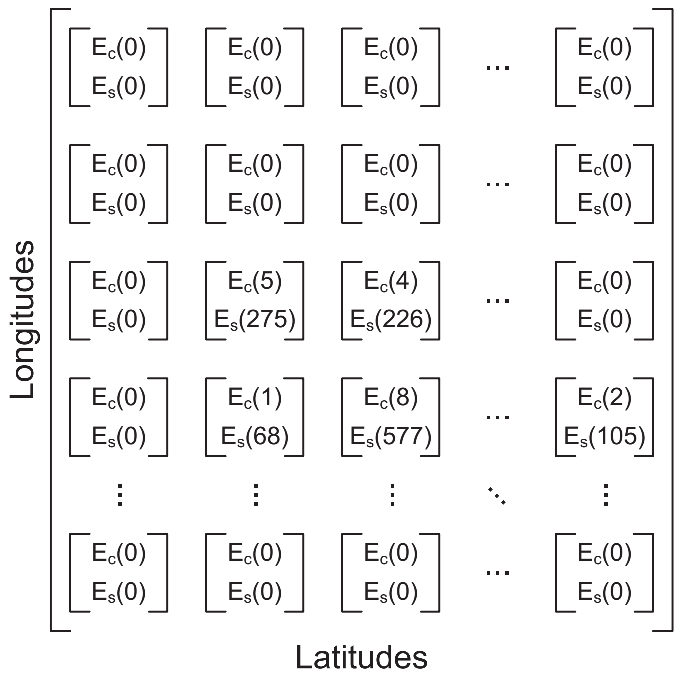


Fig. 3. A screenshot of our demo.

In case the user participates, the campaign proceeds according to the steps of Section 3. Data relevant for the campaign is encrypted at the client side in the form of a personal aggregate map using the campaign's designated public key. The structure of each personal aggregate map is shown in Fig. 3. Note that, since the map encodes the whole geographical area of the campaign, not only the sub-area the user traversed, no information can be derived on location traces. Each grid element corresponds to an area for which two values are computed: the number of measurements in the particular area (E_c), and the sum of measurements (E_s),both encrypted with the campaign's public key. By the announced deadline, each NoiseTubePrime agent has received the encrypted personal aggregate map of its user (as in Fig. 3) in case connectivity was possible with the mobile device.

When the computation deadline has been met, the distributed computation between the participating personal agents can start. Initially, the NoiseTube service selects one of the participating agents as the root-node and sends it a request to commence the map computation. Then, the root-node agent begins the computation. The computation is performed across the agent topology which provides a virtual distributed computation platform. Each agent receives the aggregate map from its predecessor and multiplies each value pair (E_s,E_c) with its own corresponding value pair. Then the result is forwarded to the successor agent in the topology, which repeats the same steps. This computation exploits the homomorphic property of the Paillier cryptosystem, which is an asymmetric cryptographic algorithm for public key cryptography (see Section 4.3 for details). Fig. 4 presents a simple ring topology, and illustrates how the computation responsibility is passed from each agent to its successor.

At the end of the computation, the aggregate encrypted map is returned to the root-node which then forwards it to the NoiseTube service for decryption. The NoiseTube service receives the aggregate map, decrypts it with the private key, and calculates the measurement average for each grid element by dividing $D(E_s)/D(E_c)$. This produces the decrypted aggregate noise map, where for each element of the grid we have calculated the average noise value and the number of measurements that support it.

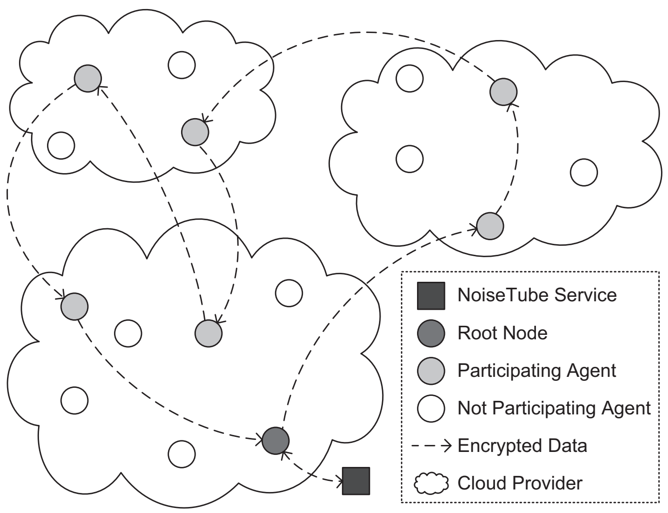


Fig. 4. Execution times of encryption by the mobile client running on a Samsung Galaxy Note II.

To avoid *side-channel privacy leaks*,[8] a user can participate even without having data for a particular computation, by submitting a private encrypted map of zero values. In this way, not even his/her own agent is aware of the fact that the user does not have data for the particular computation. Similarly, when the mobile device cannot establish contact with the NoiseTubePrime agent, the agent may participate in the computation with a private encrypted map of zero values. In this way, the agent does not need to opt out from the ring, while the final result is the same and at the same time the privacy of its owner is protected.

The appropriate network topology depends on several factors like the number of participating agents, the requirements for tolerance of network failures and the limitations on the execution time. However, in this work execution time was not critical since maps need not be computed in real-time, and our experiments with several cloud services turned out to be fast enough (see Section 5). Consequently, we adopted a simple ring topology (Lynch, 1996) – rather than, for example, a potentially faster tree-like topology – and did not investigate this issue any further.

Our protocol ensures k-anonymity (see Definition 5 in Section 4.4), where $k = N$ and N is the number of all participants that took part in the computation. Furthermore, the system could also support more statistical functions, such as covariance or frequency distribution for each grid element. Such capabilities are presented for example in Drosatos and Efraimidis (2011).

With respect to fault tolerance, during the experimental evaluation the NoiseTubePrime system was remarkably stable and reliable, even when the distributed computation comprised cloud agents from three or four different cloud providers. The only variations noticed were that some agents of a specific provider occasionally needed a longer time to wake-up from their idle state; this issue was easily addressed by executing a wake-up round before the main computation. Overall, the behaviour of the cloud services used by the agents was very reliable. This is probably not a surprise, due to the high availability of the cloud platforms provided by major players of the information technology field, like Amazon and Google. Nevertheless, a production-ready version of NoiseTubePrime should have some extra fault-tolerance features. For example, the directory server could simply skip a node of the

[8] Side-channel information leaks are information leaks that an adversary can obtain from the attributes of encrypted communications. Such side-channel leaks have been studied by (Chen et al., 2010) for example.

logical ring topology if it does not respond within a predetermined time interval to its predecessor. We do not further elaborate on such issues related to implementation improvements, since the focus of this work is on privacy-preservation rather than on network efficiency.

4.3. Cryptographic tools

The *Paillier cryptosystem* is a probabilistic asymmetric cryptographic algorithm for public key cryptography. Its security is implied by the Decisional Composite Residuosity Assumption (DCRA) (Paillier, 1999).

Definition 2 *(Homomorphic encryption). Homomorphic encryption is a form of encryption where one can perform a specific algebraic operation on the plaintext by performing a (possibly different) algebraic operation on the ciphertext.*

In NoiseTubePrime, we use the additive homomorphic encryption (Gentry, 2009; Rivest et al., 1978) property of the Paillier cryptosystem for calculating aggregate data in a privacy-preserving manner. The additive homomorphic encryption property of the Paillier cryptosystem means that multiplication of encrypted values corresponds to addition of decrypted ones. Concretely, let x_1 and x_2 be two plain integers, $x_1, x_2 \in \mathbb{Z}_{n_p}$, and (n_p, g) the Paillier public key. If r_1 and r_2 are two random numbers such that $r_1, r_2 \in \mathbb{Z}^*_{n_p}$, then the encryption of message m is $\varepsilon(m) = g^m r^{n_p} \mod n_p^2$, and the Paillier homomorphic property holds, since:

$$\varepsilon(x_1)\cdot\varepsilon(x_2) = (g^{x_1}\cdot r_1^{n_p})\cdot(g^{x_2}\cdot r_2^{n_p}) = g^{[x_1+x_2 \mod n_p]} \cdot (r_1 r_2)^{n_p} \mod n_p^2 = \varepsilon([x_1+x_2 \mod n_p])$$

We exploit the above property to calculate the measurement sum and the number of measurements in each grid element, which, when decrypted, can be divided to calculate the average value. Note that our method would work also for other functions computable with additive operations, such as covariance or frequency distribution. The same method can be used for multiplication-based aggregation if a cryptosystem supporting the multiplicative homomorphic property is used in place of Paillier. For example, the ElGamal (Elgamal, 1985) and the RSA cryptosystems (Rivest et al., 1978 Feb) support multiplicative homomorphic encryption. Moreover, there are recent results on "somewhat"[9] homomorphic cryptosystems (Gentry, 2010), i.e., cryptosystems which support a limited number of homomorphic operations including both additive and multiplicative operations.

During the last years fully homomorphic cryptosystems supporting any number of additions and multiplications have been published, starting with the seminal work of (Gentry, 2009). However, so far fully homomorphic cryptosystems are not efficient enough to be used in practical applications like NoiseTubePrime, though one could probably use "somewhat" homomorphic cryptosystems for some appropriate functions. A discussion of the efficiency and the practical relevance of current fully homomorphic and somewhat homomorphic cryptosystems, and their applications in cloud computing, can be found in Naehrig et al. (2011).

4.4. Correctness and security

In this subsection, we demonstrate that the proposed protocol is correct and that it preserves the privacy of participants.

[9] The term "somewhat" is used by (Gentry, 2010) himself to refer to an encryption scheme that can support a limited number of arithmetic operations on the encrypted data before the accumulated noise makes the resulting ciphertext indecipherable.

Definition 3 *(Correctness). A privacy-preserving extension of an algorithm is correct when it computes the exact same function as the original algorithm.*

Correctness of the PrivNoise distributed protocol follows directly from the homomorpic property of the Paillier crypto system and the additivity of computing averages. Indeed for N users $u_1, \ldots, u_N$, where each user u_i has sum of measurements s_i^{jk} and number of measurements n_i^{jk} for grid cell jk, we find that the value at the end of the computation for each grid cell is given by

$$\frac{D(E(s_1^{jk})\ldots E(s_n^{jk}))}{D(E(n_1^{jk})\ldots E(n_N^{jk}))} = \frac{D(E(s_1^{jk}+\ldots+s_n^{jk}))}{D(E(n_1^{jk})+\ldots+n_N^{jk}))} = \frac{s_1^{jk}+\ldots+s_n^{jk}}{n_1^{jk}+\ldots+n_N^{jk}} = \frac{s_{total}^{jk}}{n_{total}^{jk}}$$

which is the average sound level of grid cell jk, as required.

The security of the PrivNoise protocol holds for the *Honest-But-Curious* (HBC) model (Acquisti et al., 2008) both for the users and for the cloud providers.

Definition 4 *(Honest-But-Curious). An honest-but-curious (HBC) party (adversary) follows the prescribed computation protocol properly, but may keep intermediate computation results, e.g. messages exchanged, and try to deduce additional information from them other than the protocol result.*

In the NoiseTubePrime protocol, the information exchanged by agents is both aggregated and encrypted; thus, an honest-but-curious party cannot infer any private information. The security of the Paillier cryptosystem and its homomorphic property ensures that the personal data is not disclosed and cannot be associated with any particular user. To prove the privacy attribute of the protocol, we show that it satisfies the criterion of k-anonymity (Ciriani et al., 2007).

Definition 5 *(k-anonymity). An simple definition of k-anonymity in the context of this work is that no less than k individual users can be associated with a particular measurement value.*

The NoiseTubePrime protocol offers N-anonymity in the sense that the result computed at the end of the protocol cannot be attributed to any of the N participating agents, even if the list of participating users is known.

To summarize, the key security features of NoiseTubePrime protocol are:

- Each NoiseTubePrime agent receives an encrypted grid from the previous node. It cannot obtain information about the contents of the map, because the ciphertexts are encrypted with Paillier encryption.
- None of the cloud providers can obtain any information about the private content stored or computed by the agents, because all data and computations are in encrypted form.
- Each node alters the ciphertexts of the computation. Even the nodes that do not have data to participate multiply the ciphertexts with an encrypted number '0', which is the neutral element of the additive homomorphic property of Paillier. Again it is impossible to detect that an agent contributed with a grid consisting only of zeros.
- At the end of the protocol, only the aggregate noise map is revealed. As a result, no individual can be associated with his/her own measurements contributed in the computation. Consequently, the proposed protocol preserves k-anonymity for $k = N$, where N is the number of all participants that took part in the computation.

Our protocol can be extended to tolerate (at least some types of) malicious behaviour. For example, a malicious NoiseTube service could collude with potential malicious cloud providers or user agents to obtain and decrypt intermediate results of the computation. This could possibly lead to the disclosure of the personal maps submitted by specific users. Such a threat can be effectively handled by deploying threshold decryption (Damgård and Jurik, 2001) for the decryption of the encrypted maps. Threshold decryption requires that the number of coordinating parties exceeds an appropriate threshold for decryption to be possible. We leave the comprehensive treatment of malicious user behaviour within our application for future work.

5. Experimental evaluation

To evaluate our approach, we developed a NoiseTubePrime prototype that implements the proposed privacy-preserving protocol for calculating participatory noise maps in the cloud. We used the implementation to set up an online demo of a NoiseTubePrime use case and to execute two sets of experiments for privacy-preserving noise map generation, showing that our protocol is able to reproduce noise maps correctly. We also analyze the performance of our protocol, in the context of real as well as artificial setups.

5.1. The NoiseTubePrime prototype

The prototype consists of two parts: the mobile application, which runs on users' devices, and the NoiseTubePrime agent community, which runs on (a family of) cloud providers.

At the mobile device side, we implemented our solution on the Android platform,[10] using Java. We have chosen Android because the existing NoiseTube system already supports it, and because it is currently the most popular smartphone platform.[11] However, there is no reason why our solution could not be ported to other mobile application platforms (e.g. Java ME/CLDC or Apple iOS). For convenience, our implementation uses the BigInteger class provided by Android (and by Java SE, but not Java ME/CLDC), but on platforms that do not provide a similar type or class this could be implemented at the level of the application itself. The NoiseTubePrime agents were also implemented in Java, as Java Web Servlets (WAR). They were deployed on several cloud infrastructure providers, namely Google App Engine, CloudBees, and Amazon EC2, without important differences in the implementation.[12]

In the current stage of their development, NoiseTubePrime agents and the Android client application do not have all functionalities that were presented in the previous sections – in particular those parts pertaining to campaign definitions are currently lacking. However, we did fully implement and test the core of the protocol, i.e., the distributed homomorphic computations in a realistic setting. Our prototype supports both http and https for the communication among NoiseTubePrime agents and between the Android client and the cloud agent. The https protocol, which makes use of encrypted communication over secure sockets (SSL/TLS), is necessary to fully satisfy the security goals of NoiseTubePrime. In our experiments, however, for simplicity[13] we used http. Both the mobile and the cloud application implement the Paillier cryptosystem primitives for encrypting/decrypting data and performing secure calculations.

[10] In fact, we are targeting Android v2.2 "Froyo", or newer versions.

[11] Android runs on almost 80% of smartphones sold in Q2 2013 (Gartner Inc., 2013).

[12] *Google App Engine*: http://appengine.google.com, *CloudBees:* http://www.cloudbees.com, *Amazon EC2:* http://aws.amazon.com/ec2.

[13] The configuration of https was a provider-specific task, which was complicated for some of the providers.

5.2. On-line demonstration

To demonstrate NoiseTubePrime functionality, we implemented an online demo (http://polis.ee.duth.gr/NoiseTubePrime), for a small scale experiment. As a proof of concept and at the same time a validation of correctness, our demo reproduces, in a privacy-preserving way, the results of a concrete noise measuring campaign. For this purpose, we used real noise measurements collected in July 2010 by volunteering citizens in a 0.4 km × 0.4 km area in the city of Antwerp in Belgium, as part of the "Ademloos experiment" set up by the BrusSense Team (D'Hondt et al., 2013; Stevens, 2012).

Concretely, the campaign's goal was to map the chosen area during a peak-hour (7:30–8:30 am) and an off-peak hour (9:00–10:00 pm). To do this, four volunteers from the Antwerp-based Ademloos citizen action group followed a pre-defined measurement track twice daily for a week for each of the chosen hours. On the basis of these measurements (over 30,000 for each week) noise maps of the target area were produced. The standard NoiseTube approach is to analyze a collection of measurement tracks statistically to produce one single noise map. To do this the measured area is divided into smaller areas, the total set of measurements is divided over those areas, and a statistical analysis is carried out per unit area. In a final step colour coded averages are mapped on each pertaining area. The resulting noise maps for this and other experiments can be found online.[14]

In our online demo, we deploy four NoiseTubePrime agents which represent the four volunteers of the Ademloos experiment, and re-compute the same maps in a privacy-preserving manner using the NoiseTubePrime protocol. The demo is implemented with the Google Web Toolkit (GWT v2.4.0)[15] and consists both of a Web client and a server side application (servlet). The Web client is used to control the four mobile clients of the demo and visualizes the final results. The servlet initiates the computation, so that the four agents compute the aggregate map from the encrypted data of their users. Moreover, for the particular demo the servlet is used to simulate the four mobile devices. For this purpose, the original Java classes implementing the computational task of the Android application have been packaged within the server side servlet. The four agents have been tested on three different cloud providers (Google App Engine, CloudBees, and Amazon EC2). The key size of Paillier cryptosystem was chosen to be 512 bits. A screenshot of the demo during the execution of an experiment is shown in Fig. 5. Each grid element corresponds to an area of 40 m × 40 m.

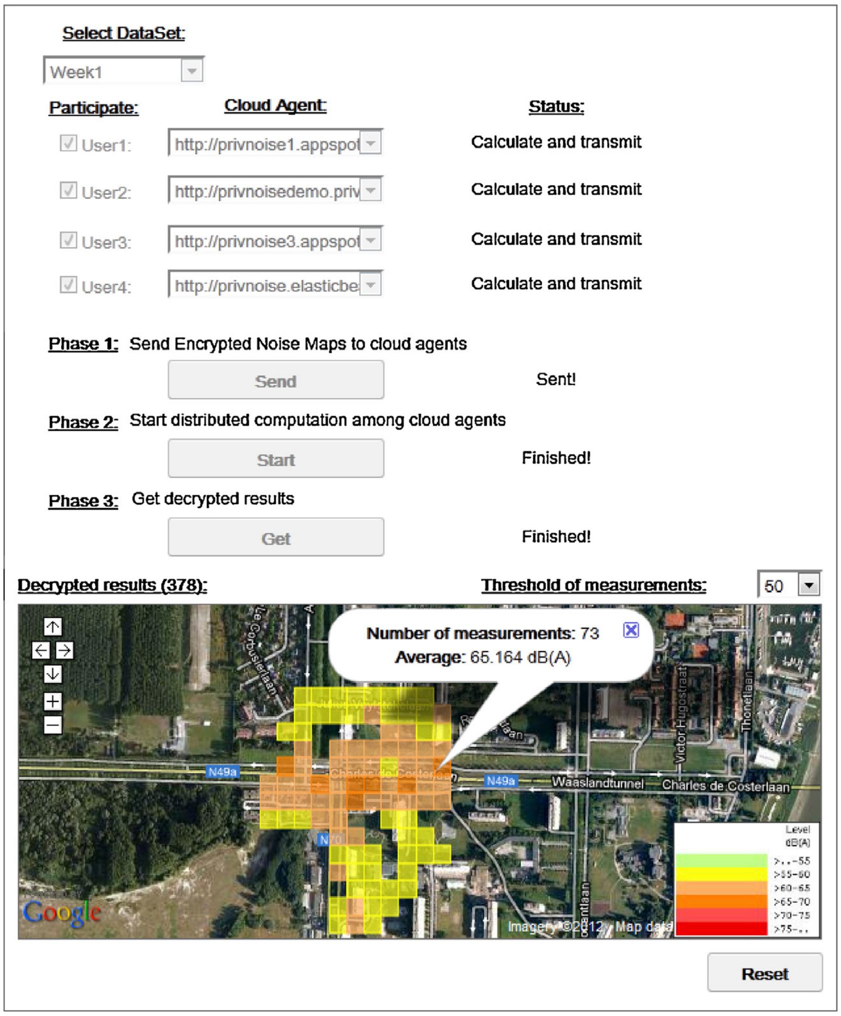
The time needed by the NoiseTubePrime cloud agents for the multiplication of the encrypted maps, fluctuates between 875 and 1614 ms, which is acceptable for a grid of 21 × 18 elements. Note that this includes the time for receiving and transmitting the aggregate encrypted map, because we had no simple way to separate these two quantities from the cloud providers logs.

5.3. Computational performance evaluation

To evaluate the computational requirements of the NoiseTubePrime system for a wide range of realistic problem sizes and security parameters we conducted a large set of experiments using noise data which was generated artificially rather than actually measured. These comprised performance evaluation both of the mobile device-based computation and the distributed cloud-based computations. Naturally, the location trace of each simulated user and the number and the values of the corresponding noise measurements have only negligible impact on the computational

[14] http://www.brussense.be/experiments/.

[15] http://code.google.com/webtoolkit/.


Select DataSet:
Week1
Participate:	Cloud Agent:	Status:
✓ User1:	http://privnoise1.appspot	Calculate and transmit
✓ User2:	http://privnoisedemo.priv	Calculate and transmit
✓ User3:	http://privnoise3.appspot	Calculate and transmit
✓ User4:	http://privnoise.elasticbe	Calculate and transmit

Phase 1: Send Encrypted Noise Maps to cloud agents — Send — Sent!
Phase 2: Start distributed computation among cloud agents — Start — Finished!
Phase 3: Get decrypted results — Get — Finished!
Decrypted results (378): Threshold of measurements: 50
Number of measurements: 73
Average: 65.164 dB(A)

Fig. 5. Execution times of encryption by the mobile client running on different mobile devices.

requirement of the NoiseTubePrime application. Instead, the running time of the application is dominated by the size of the noise map, which is determined by the number of grid elements, and the size of the encryption keys. Thus, the computational requirements for processing the artificial data closely resembles the corresponding task on real data.

The NoiseTubePrime solution comprises two main computational tasks (given that the noise measurements are collected through the existing NoiseTube application): The preprocessing step (encryption of the local map, step (c) in Section 3) which is executed locally on the mobile devices, and the distributed computation step (merging of the encrypted maps, step (f) in Section 3) which is outsourced to the personal agents located in the Cloud.

The first set of experiments concerns the computation task of the mobile devices, which have to prepare the encrypted map for each user. This task is highly parallelizable, and thus we can fully exploit the multi-core CPU architectures of modern mobile devices. In Fig. 6 we show the execution times of the data encryption step that is performed by a mobile device for different map and public key sizes. In this experiment, we used the Android smartphone Samsung Galaxy Note II, which comes with a quad-core CPU ARM Cortex-A9 at 1.6 GHz and 2 GB of RAM. While execution times may run up to a few minutes, in particular for large maps, we note that the running time of this task is not critical for the NoiseTubePrime application, since it can be executed in batch mode as a background task on the mobile device at any time before the deadline of the specific campaign. It would also be possible to completely hide the running time of this preprocessing task, for example by incrementally building the local encrypted map during the noise sampling

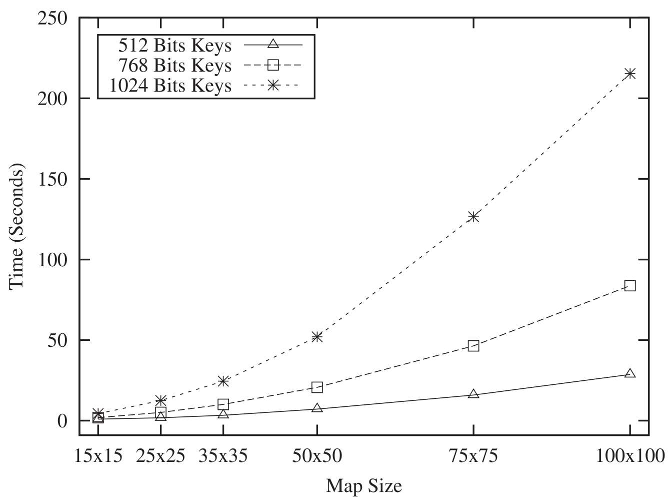


Fig. 6. Execution times of computation in the cloud with artificially generated noise data.

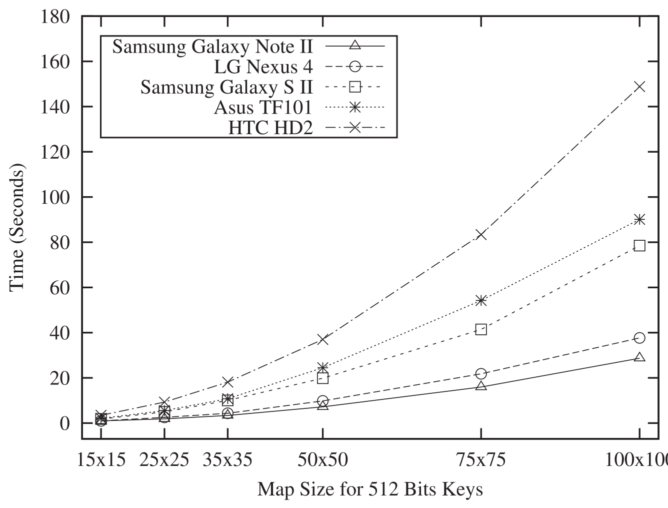


Fig. 7. Execution times of computation in the cloud with real noise measurements.

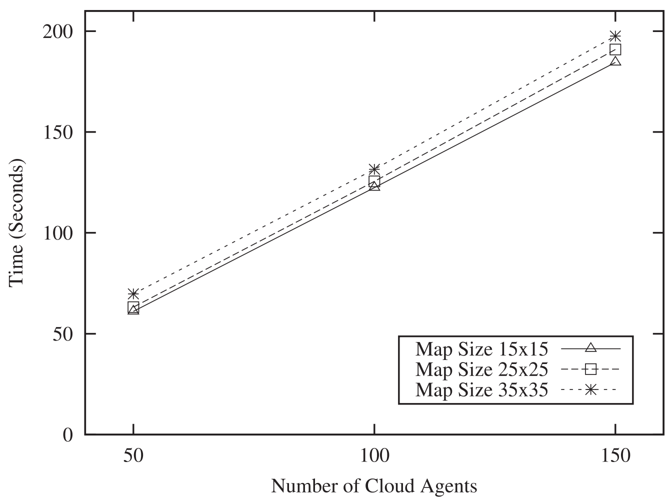


Fig. 8. Execution times of computation in the cloud with artificially generated noise data.

phase; a background process of the mobile device could immediately encrypt every new measurement and merge it with the local encrypted map.

Next we examine how the execution time of the mobile application may vary between different mobile devices. In Fig. 7 we show the execution time of the data encryption step that is performed on five different modern mobile devices, a Samsung Galaxy Note II (4-core CPU), a LG Nexus 4 (4-core CPU), a Samsung Galaxy S II (2-core CPU), an Asus TF101 (2-core CPU) and an HTC HD2 (single-core CPU). In these experiments the key size of the Paillier cryptosystem was fixed at 512 bits. Execution times do not vary substantially, at least not for the chosen set of devices. However, we do clearly see the effect of the parallel nature of the problem by the difference in single, dual and quad-core curves.

With respect to the Web-based component, Fig. 8 shows how the execution time of the distributed computation in the cloud varies with respect to the number of NoiseTubePrime agents. In this simulation we deployed agents on a single cloud provider (Google App Engine), in a simple ring topology and with 512 bits Paillier keys. We use a single cloud provider in this set of experiments so as to minimize delays due to network transmission and as a result, we mainly capture the processing time of the cloud agents. Fig. 8 shows that the total time for the distributed computation increases almost linearly with the number of agents, while the size (number of cells) of the map has only a small impact on the running time. In particular, for map sizes 15×15, 25×25 and 35×35, the delay per node is approximately 1.22, 1.26 and 1.31 s, respectively.

We consider the execution times in all the above experiments to be entirely acceptable for noise mapping campaigns. There are of course feasible ways to improve the computational performance further, if necessary. The mobile device application can be further optimized by using more advanced programming techniques such as Renderscript (Qian et al., 2012) for the encryption process. Renderscript offers a high performance computation API for Android devices that gives the ability to run operations with automatic parallelization across all available processor cores of a device such as the CPU, GPU and DSP. On the other hand, the execution time of the cloud agent computation can be reduced by using a more efficient virtual topology, which would increase the concurrency of the distributed computation, by requesting more powerful resources from the cloud providers and by compressing the data that is transmitted during the distributed computation. However, we repeat that because both transmission of data as well as producing the agglomerated map are not required to proceed in real-time any delays that we find do not pose a concern.

5.4. A realistic use-case of NoiseTubePrime

For our final experiment we set up what we consider to be a realistic use case of noise mapping and the NoiseTubePrime application. In this experiment we rely on a data set of real noise measurements gathered by 93 users in a 4 km^2 area in the city of Brussels, Belgium. The data set comprises 409,768 measurements at an average of 4406 measurements per user, gathered in an uncoordinated way over a long period of time and including calibrated as well as uncalibrated devices. The largest user contribution consists of 76,337 and the smallest of 12 measurements.[16]

With respect to privacy, we assume a mixed environment specified by user preference. In this way we consider two user types: “conventional” NoiseTube users, who contribute their data in plain format, and privacy-sensitive NoiseTubePrime users, who wish to share data only in a privacy-preserving way. Based on this assumption, we conducted a realistic set of experiments with a varying number of privacy-sensitive users.

With respect to the cloud platforms, we assumed heterogeneity too. Agents of the privacy-sensitive users run on three possible platforms: two commercial cloud providers, *CloudBees* and *Google App Engine*, and a server running in our lab. We performed a series of experiments with a gradually increasing number (up to 40 out of a total of 93) of privacy-aware users that deploy NoiseTubePrime agents, while the remaining users participate in the campaign as conventional NoiseTube users. In all experiments, the numbers of the agents assigned to each provider satisfy the ratio 1:1:2 for Google App Engine, CloudBees and the own server, respectively. For example, in the case of 40 privacy-aware users, we deployed 10 agents on the *Google App Engine*, 10 agents on *CloudBees*, and 20 agents on our own server.

We used a simple ring topology where the sequence of agents alternated between those residing on a cloud provider and on our own server. Note that this sequence of agents corresponds to a worst-case scenario with respect to the network load, since it generates the maximum possible network traffic for the particular mixture of agents.

Fig. 9 shows how the execution time of the distributed computation varies with respect to the number of NoiseTubePrime agents

[16] Because of the heterogeneous nature of this dataset we have no guarantee about the quality of the resulting noise map and therefore choose not to include it here. However, that does not affect the usefulness of this dataset for the purpose of evaluating the NoiseTubePrime privacy-preserving map computation system.

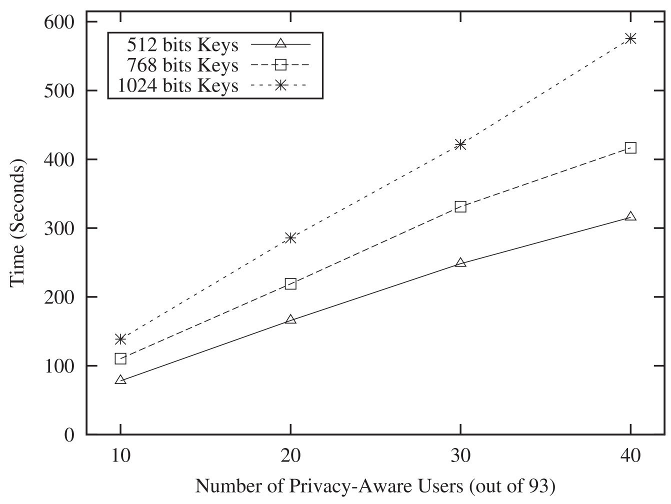


Fig. 9. Execution times of computation in the cloud with real noise measurements.

and public key sizes. The map size is 100 × 100 elements and each grid element corresponds to an area of 40 m × 40 m. In each experiment we also verified that the final aggregated noise map of the privacy-preserving and the conventionally computed results were identical.

We again find computation times which evolve linearly with the number of cloud agents. Moreover computation times are of a duration that is perfectly acceptable for a map of this size. Indeed, even without a privacy-preserving computation producing a noise map for this amount of measurements typically takes up a couple of minutes, and moreover we stress once more that producing noise maps is not something that is required to happen in real-time.

6. Related work/discussion

6.1. On privacy-preserving participatory sensing

Privacy protection in mobile sensing systems has recently attracted the interest of the scientific community (Lane et al., 2010; Christin et al., 2011). Because users of a participatory sensing system play an active role in the data collection process, it has been argued that they should also be actively engaged in privacy-related decisions (Shilton, 2009), e.g. where and when to measure and what to share with whom. It has also been argued that, in order to protect user privacy and increase their negotiating power, data collection and data sharing should be decoupled by introducing a *personal data vault* that stores a user's data in a secure manner (i.e. encrypted), from which he/she can then selectively share subsets with various services or campaigns (Estrin, 2010). This idea is one of the ingredients of the NoiseTubePrime system presented above.

In Christin et al. (2011) a comprehensive survey on related work as well as important challenges are covered. Most related work discussed in this survey and beyond uses a fundamentally different approach than the one advocated here, as is detailed further in Table 1. Often, sensor data integrity or accuracy is sacrificed for privacy preservation. For example, the approach presented in Kapadia et al. (2008) divides the area under consideration into appropriate regions (tessellation procedure), which have to be sufficiently large to preserve user anonymity. A similar approach is used in Chow et al. (2011), where area cloaking is used to offer k-anonymity. The NoiseTubePrime approach is simpler and does not require any specific area division to preserve user privacy.

A very interesting related work is the PriSense system (Shi et al., 2010) which is based on a P2P data slicing technique (He et al., 2007), and can offer functionality comparable to NoiseTubePrime

Table 1
Comparison with related work.

Name	(i) Approach	(ii) Application domain	(iii) Validation	(iv) # users	(v) Correctness	(vi) Cloud	(vii) Implementation
NoiseTubePrime	Homomorphic encryption, only local storage of raw data, distributed topology	General purpose for additive data aggregation in people-centric urban sensing	Online demo, simulation with real and artificially generated data	Hundreds	Yes	Yes (variety of real cloud providers)	Android app, Java web servlets, simulation
AnonySense (Kapadia et al., 2008)	Tessellation and clustering algorithm	General purpose for anonymous tasking and reporting in people-centric sensing systems	Simulation with real data, real mobility traces	Thousands	Unclear	No	Simulation
PoolView (Ganti et al., 2008)	Data perturbation	Grassroots participatory sensing	Data from real and emulated users	Hundreds	Approximate	No	Web Server application, Perl, SciLab
PriSense (Shi et al., 2010)	Data slicing and mixing	General purpose for data aggregation in people-centric urban sensing systems	Theoretical approach, mathematical model	N/A	Approximate statistical results	No	No
(Becchetti et al., 2010)	Sketches	Opportunistic monitoring	Simulated mobility traces	Hundreds	Approximate data representation (sketches)	No	Simulation
LOCATE (Boutsis and Kalogeraki, 2013)	MapReduce framework for mobile devices in distributed topology	Participatory sensing in general	Real GPS trajectories	Hundreds	Yes	No	Android app
(Krontiris and Dimitriou, 2013)	Obfuscation, distributed topology	Participatory sensing mapping with selection criteria on specific mobile nodes	Simulated set of moving objects	Thousands	Yes	Yes	Simulation

for additive aggregation functions. However, the homomorphic encryption-based approach of NoiseTubePrime is simpler – no data scattering has to take place – and seems to be more general since homomorphic encryption is not limited to additive functions. Moreover, PriSense only ensures data privacy protection if the nodes and the server do not conspire to breach the privacy of potential targets (Christin et al., 2011).

The work in Becchetti et al. (2010) uses advanced algorithmic techniques like sketches and approximate set cover to compute approximate statistic results in a privacy-preserving way. While theoretically interesting, in our opinion this approach is too complicated to be applied in practical, real-world settings, as opposed to NoiseTubePrime.

With respect to existing work, NoiseTubePrime is a simple but at the same time powerful approach for privacy-preserving participatory sensing, and to the best of our knowledge, the first operational privacy-preserving solution for participatory sensing. Moreover, NoiseTubePrime minimizes the requirements for the user by outsourcing the distributed computation to free (or very cheap) cloud computing services.[17] NoiseTubePrime is based on plausible assumptions, is efficient for our purposes (as is shown in Section 5), has no requirements for special infrastructure and does not make any compromises in the quality of the computed results like cloaking or tessellation do. Similarly to PriSense, NoiseTubePrime is user-centred but, unlike PriSense, it can also support multiplicative functions by using an appropriate homomorphic cryptosystem.

6.2. *On using cloud agents for preserving privacy*

The NoiseTubePrime software agents that we introduce in this work are based on the related idea of the Polis Project (Efraimidis et al., 2009) where each user is represented by a Polis agent. In a nutshell, Polis is a personal data management framework that abides by the following principle: every individual has absolute control over his/her personal data that reside only at his/her own side. The Polis agents constitute the backbone of the Polis architecture and run on the user side; they are used to manage the personal data of a user, and provide controlled access at the entity's data. The service providers request personal data items of users from their personal agents. The agents provide the requested data if there is a corresponding license agreement (policies).

NoiseTubePrime agents are deployed in the cloud as Web services and are located on public servers, in contrast with Polis agents that are on the users' side. To avoid the obvious disclosure of personal data to cloud providers, only data in encrypted form is uploaded to the agents. The encryption of data solves both security and privacy issues that we have in clouds. Furthermore, NoiseTubePrime agents host only data that is destined for particular computations, and not all personal data, such as Personal Identifiable Information (PII), of users. For common security requirements like authentication of users who have the right to add personal encrypted data and to participate in a specific distributed computation, we can use standard security measures. Finally, the main reasons why we deploy our agents in the cloud are:

- Agents have to be online continuously during the distributed computation;
- Several cloud computing providers offer free services for low computational and bandwidth requirements, which are sufficient for our goals;
- The network connectivity offered by cloud infrastructures is fast and reliable, unlike mobile data connections;
- The cloud offers scalable computational resources.

It is noteworthy that the cloud agent approach first presented in the conference version of this work (Drosatos et al., 2012) has since been adopted in a very recent work of (Krontiris and Dimitriou, 2013) for a solution achieving a different privacy goal (privacy of the query) in the context of crowd-sensing applications.

7. Conclusion

This paper presents a novel, privacy-preserving architecture for the creation of participatory noise maps, called NoiseTubePrime and built on top of the NoiseTube system (Maisonneuve et al., 2010; Stevens, 2012). NoiseTubePrime allows aggregate noise maps to be generated from data collected by multiple users without disclosing their location traces. The protocol is correct in that the resulting maps are exactly the same as those generated with conventional grid-based aggregation methods, as applied in D'Hondt et al. (2013) and Stevens (2012). However, our system allows users to preserve their privacy, and thus contributes to the realization of trustworthy computing systems. Our approach implements the '*fair information principle*' as privacy is respected when information is collected (Mundie et al., 2002). The protection of privacy is achieved by using cryptographic techniques and performing a distributed computation within a network of software agents. The distributed computation is performed on encrypted data and no personal information is disclosed to anyone, including the cloud service providers, at any time. Finally, we developed a prototype implementation and presented experimental results using a heterogeneous set of commercial cloud services, confirming the viability and the efficiency of the proposed solution. Key features of the NoiseTubePrime system include:

- Correctness: Accurate aggregate statistics are computed using the private measurement data of each user, while at the same time the privacy of the participating users is preserved: No location/time data is disclosed.
- Cloud services: Outsourcing the NoiseTubePrime agent to the cloud relieves the user from the trouble to run and manage his/her own software agent and to maintain permanent Internet access. The computational and networking requirements of each software agent are low and are (currently) provided without any cost by the various cloud service providers we used in our experiments.
- Decentralization: The main task of the NoiseTube service is to decrypt the final encrypted noise map that is the result of the distributed cloud computation. Thus, the central workload is much lower than in a scenario where the whole computation is performed on a a single central server. Hence the computational work of the NoiseTube service is independent of the number of participating users, making NoiseTubePrime a decentralized system that theoretically can be scaled to handle very large numbers of users.

The NoiseTubePrime architecture strikes a sound balance between providing secure, yet straightforward, privacy protection for those contributors that want or require it, while maintaining transparency for those that do not. We believe that the privacy-preserving solution presented in this work can make participatory sensing platforms like NoiseTube more suitable for medium- to

[17] The requirements for computation and networking per user are very low and therefore the free, but limited, šstarterþackages offered by several cloud providers suffice to run NoiseTubePrime agents. In the foreseeable future we see no reason why such services would no longer be offered for free or, in the worst case, at a very low cost.

large-scale (e.g. city-wide) deployments, in which the privacy concerns of individual contributors are expected to be significantly higher than in previous small-scale noise mapping campaigns (Stevens, 2012; D'Hondt et al., 2013) – due to higher numbers of participants, weaker (or absent) acquaintance and trust relationships, and possibly the involvement of authorities.

Our future plans are to develop a stable and more complete version of NoiseTubePrime and demonstrate its use for real-world campaigns, also extending the platform towards more statistical parameters. To accomplish this we have to extend our prototype implementation. Roughly, the user-side Android application has to be to enriched with features supporting user policies and campaign participation and the resulting code has to be integrated into the existing NoiseTube Mobile for Android application,[18] and then released to the public. The prototype NoiseTubePrime servlet, which implements the server side of our application, has to be extended with auxiliary functionalities for public key management, campaign management and a Directory Service for supporting the distributed computations.[19] Current and future NoiseTube users should be oblivious to these privacy extensions insofar as possible. In the future a user study could be set up to evaluate the overall usability of the solution in different contexts.

Last but not least we should stress that the proposed architecture for privacy-preserving sharing, transmission, processing and management of sensitive (spatial) data is independent of the noise domain, and can thus potentially be applied in other participatory sensing systems. The only constraint is that the parameters of interest can be computed with efficient homomorphic cryptosystems.

Acknowledgements

G. Drosatos and P.S. Efraimidis were partially supported from the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 264226: SPace Internetworking CEnter – SPICE. This paper reflects only the views of the authors – The Union is not liable for any use that may be made of the information contained. G. Drosatos, P.S. Efraimidis and I.N. Athanasiadis are partially supported by ETAA funds. E. D'Hondt is supported by the InnovIris, the Brussels Institute for Research and Innovation, and the EU-CIP i-Scope project. M. Stevens is currently supported by the EPSRC, under the 'Extreme' Citizen Science grant (EP/I025278/1).

References

Acquisti, A., Gritzalis, S., Lambrinoudakis, C., De Capitani di Vimercati, S., 2008. Digital Privacy. Auerbach Publications, Taylor & Francis Group, Broken Sound ParkWay NW.

Becchetti, L., Filipponi, L., Vitaletti, A., 2010. Opportunistic privacy preserving monitoring. In: PhoneSense '10: International Workshop on Sensing for App Phones held at ACM SenSys, vol. 10, pp. 51–55.

Bilogrevic, I., Jadliwala, M., Kumar, P., Walia, S.S., Hubaux, J.-P., Aad, I., Niemi, V., 2011. Meetings through the cloud: privacy-preserving scheduling on mobile devices. Journal of Systems and Software 84 (11), 1910–1927.

Boutsis, I., Kalogeraki, V.,2013. Privacy preservation for participatory sensing data. In: Proceedings of the 11th IEEE International Conference on Pervasive Computing and Communications (PerCom '13). IEEE Computer Society, Los Alamitos, CA, pp. 103–113.

Burke, J.A., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., Srivastava, M.B., 2006. Participatory sensing. In: WSW '06: Workshop on World-Sensor-Web held at ACM SenSys '06, October.

Buytendijk, F., Heiser, J., 2013. Confronting the privacy and ethical risks of big data. The Financial Times, http://on.ft.com/1dZq0C4 (24.09.13).

Chen, S., Wang, R., Wang, X., Zhang, K.,2010. Side-channel leaks in web applications: a reality today, a challenge tomorrow. In: Proceedings of the 2010 IEEE Symposium on Security and Privacy (SP '10). IEEE Computer Society, Washington, DC, pp. 191–206.

Chow, C.-Y., Mokbel, M.F., He, T., 2011. A privacy-preserving location monitoring system for wireless sensor networks. IEEE Transactions on Mobile Computing 10 (January (1)), 94–107.

Christin, D., Reinhardt, A., Kanhere, S.S., Hollick, M., 2011. A survey on privacy in mobile participatory sensing applications. Journal of Systems and Software 84 (11), 1928–1946.

Ciriani, V., Capitani di Vimercati, S., Foresti, S., Samarati, P., 2007. κ-Anonymity. In: Secure Data Management in Decentralized Systems. Vol. 33 of Advances in Information Security, Springer, pp. 323–353.

Damgård, I., Jurik, M.,2001. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography (PKC '01). Springer-Verlag, London, UK, pp. 119–136.

D'Hondt, E., Stevens, M., 2011. Participatory noise mapping. In: Ballagas, R.T., Rosner, D.K. (Eds.), Demo Proceedings of the 9th International Conference on Pervasive Computing (Pervasive '11). June, pp. 33–36.

D'Hondt, E., Stevens, M., Jacobs, A., 2013. Participatory noise mapping works! An evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring. Pervasive and Mobile Computing 9 (5), 681–694.

Drosatos, G., Efraimidis, P.,2011. Privacy-preserving statistical analysis on ubiquitous health data. In: Proceedings of the 8th International Conference on Trust, Privacy and Security in Digital Business (TrustBus '11), Vol. 6863 of LNCS. Springer, Berlin/Heidelberg, pp. 24–36.

Drosatos, G., Efraimidis, P.S., 2014. An efficient privacy-preserving solution for finding the nearest doctor. Personal and Ubiquitous Computing 18 (1), 75–90.

Drosatos, G., Efraimidis, P.S., Athanasiadis, I.N., D'Hondt, E., Stevens, M., 2012. A privacy-preserving cloud computing system for creating participatory noise maps. In: 36th Annual IEEE Computer Software and Applications Conference (COMPSAC 2012), IEEE Computer Society, July, pp. 581–586.

Efraimidis, P.S., Drosatos, G., Nalbadis, F., Tasidou, F.A., 2009. Towards privacy in personal data management. Journal on Information Management & Computer Security 17 (4), 311–329.

Elgamal, T., 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31 (July (4)), 469–472.

Estrin, D., 2010. Participatory sensing: applications and architecture. IEEE Internet Computing 14 (January/February (1)), 12–14.

Ganti, R.K., Pham, N., Tsai, Y.-E., Abdelzaher, T.F.,2008. Poolview: stream privacy for grassroots participatory sensing. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys '08). ACM, New York, NY, pp. 281–294.

Gartner Inc., 2013. Gartner says smartphone sales grew 46.5 percent in second quarter of 2013 and exceeded feature phone sales for first time, http://www.gartner.com/newsroom/id/2573415 (14.08.13).

Gentry, C.,2009. Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC '09). ACM, New York, NY, pp. 169–178.

Gentry, C., 2010. Computing arbitrary functions of encrypted data. Communications of the ACM 53 (March (3)), 97–105.

He, W., Liu, X., Nguyen, H., Nahrstedt, K., Abdelzaher, T., 2007. PDA: privacy-preserving data aggregation in wireless sensor networks. In: Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM 07), IEEE, May, pp. 2045–2053.

Kapadia, A., Triandopoulos, N., Cornelius, C., Peebles, D., Kotz, D., 2008. AnonySense: opportunistic and privacy-preserving context collection. In: Indulska, J., Patterson, D., Rodden, T., Ott, M. (Eds.), Proceedings of the 6th International Conference on Pervasive Computing (Pervasive '08) Vol. 5013 of LNCS. Springer, Berlin/Heidelberg, May, pp. 280–297.

Krontiris, I., Dimitriou, T.,2013. Privacy-respecting discovery of data providers in crowd-sensing applications. In: Proceedings of the 9th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '13). IEEE Computer Society, Los Alamitos, CA, pp. 249–257.

Lane, N.D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., Campbell, A.T., 2010. A survey of mobile phone sensing. IEEE Communications Magazine 48 (September (9)), 140–150.

Lynch, N.A., 1996. Distributed Algorithms. Morgan Kaufmann Publishers Inc., San Francisco, CA.

Maisonneuve, N., Stevens, M., Ochab, B., 2010. Participatory noise pollution monitoring using mobile phones. Information Polity 15 (August (1–2)), 51–71.

Mundie, C., de Vries, P., Haynes, P., Corwine, M., 2002. Trustworthy Computing. Microsoft white paper. Microsoft Corporation http://download.microsoft.com/download/a/f/2/af22fd56-7f19-47aa-8167-4b1d73cd3c57/twc_mundie.doc

Naehrig, M., Lauter, K., Vaikuntanathan, V.,2011. Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop (CCSW '11). ACM, New York, NY, pp. 113–124.

Paillier, P., 1999. Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology – EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques. Vol. 1592 of LNCS, Springer, pp. 223–238.

Paulos, E., 2009. Citizen science: enabling participatory urbanism. In: Foth, M. (Ed.), Handbook of Research on Urban Informatics: The Practice and Promise of the Real-Time City. Information Science Reference, IGI Global, Hershey, New York, pp. 414–436 (Chaper 28).

[18] https://play.google.com/store/apps/details?id=net.noisetube.

[19] We note that the NoiseTube system as it stands is currently undergoing a transition to support campaigns. However, the privacy extensions proposed in this article are not yet included.

Qian, X., Zhu, G., Li, X.-F, 2012. Comparison and analysis of the three programming models in google android. In: Proceedings of the 1st Asia-Pacific Programming Languages and Compilers Workshop (APPLC) in Conjunction with PLDI, June.

Rivest, R., Adleman, L., Dertouzos, M., 1978. On data banks and privacy homomorphisms. In: Foundations of Secure Computation. Academic Press, New York, pp. 169–177.

Rivest, R.L., Shamir, A., Adleman, L., 1978 Feb. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21 (February (2)), 120–126.

Shi, J., Zhang, R., Liu, Y., Zhang, Y., 2010. PriSense: privacy-preserving data aggregation in people-centric urban sensing systems. In: Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM '10), IEEE, March, pp. 1–9.

Shilton, K., 2009. Four billion little brothers? Privacy, mobile phones, and ubiquitous data collection. Communications of the ACM 52 (November), 48–53.

Steels, L., 2007, November. Community Memories for Sustainable Societies. Tech. rep., Sony Computer Science Laboratory, Paris http://csl.sony.fr/downloads/papers/2007/steels-07a.pdf

Stevens, M., (Ph.D. thesis) 2012, June. Community Memories for Sustainable Societies: The Case of Environmental Noise. Vrije Universiteit Brussel.

Streitfeld, D., Hardy, Q., 2013. Data-driven tech industry is shaken by online privacy fears. The New York Times http://nyti.ms/1cP1NRE (09.06.13).

Yao, A.C.-C.,1982. Protocols for secure computations (extended abstract). In: Proceedings of Twenty-third IEEE Symposium on Foundations of Computer Science (FOCS 'N82). IEEE, Los Alamitos, Chicago, IL, November, pp. 160–164.