Managing Personal Health Records in an Infrastructure-Weak Environment

Nicolas Anciaux[1(✉)], Sébastien Guillotton[2], Luc Bouganim[1], Sergio Ilarri[3], Alain Kamgang[4], Abraham Ngami[5], Christophe Nouedoui[4], Philippe Pucheral[6], and Maurice Tchuente[5]

[1] Inria, Rocquencourt, France
nicolas.anciaux@inria.fr
[2] Ensta, Palaiseau, France
[3] University of Zaragoza, Saragossa, Spain
[4] Hospital Central, Yaoundé, Cameroon
[5] University of Yaoundé, Yaoundé, Cameroon
[6] University Versailles Saint-Quentin-en-Yvelines, Versailles, France

Abstract. There are currently more than half a million diabetes cases in Cameroon and the deaths caused by diabetes complications will double before 2030. Diabetes complications mostly occur due to a bad follow-up of patients. In this paper, we propose a new IT architecture for diabetes follow-up and introduce the bases of a new distributed computation protocol for this architecture. Our approach does not require any preexisting support communication infrastructure, can be deployed at low cost, and provides strong privacy and security guarantees. This work envisions an experiment in the field we plan to conduct under the authority of the Cameroonian National Center for Diabetes and Hypertension, with a potential for generalization to other diseases.

1 Introduction

According to the World Health Organization, 347 million people worldwide have diabetes [1]. This is a chronic and incurable disease, for which good treatments do exist. Diabetes is however currently the direct cause of 1.5 million deaths [2] with more than 80 % of deaths in low and middle income countries [5] and deaths caused by diabetes are expected to double before 2030 [4].

Diabetes follow-up plays an important role to orchestrate the treatments, which are indeed a combination of insulin drugs, appropriate and healthy diet, regular physical activities, and patient monitoring. An incorrect follow-up of patients may cause hyperglycemia, which over time would provoke an alteration of the nerves and blood vessels with many bad effects on the body. These diabetes complications might cause severe problems like hypertension, blindness, feet problems leading to amputations, kidneys failure, and myocardial or heart stroke with significant morbidity. Fortunately, they can be avoided if there is a good follow-up of diabetes patients. Our discussions with doctors from the *Cameroonian National Center for Diabetes and Hypertension* (NCDH) have led to three main requirements to improve patients follow-up: (1) Patients should own a Personal Health Record (PHR) and make it available to the

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2016
R. Glitho et al. (Eds.): AFRICOMM 2015, LNICST 171, pp. 178–191, 2016.
DOI: 10.1007/978-3-319-43696-8_18

practitioners at the time of the consultation; (2) global statistics on populations of patients should be computed easily for a better understanding of diabetes evolution, its complications and co-morbidities, in order to adapt prevention actions and treatments to local populations; and (3) a synthetic view of the PHR of each patient could be maintained on a server in the different services or hospitals. We concentrate in this paper on the first two points and let the third one for future work. It should be noted that the benefit of a PHR even goes beyond the case of diabetes [9].

The use of e-services is unavoidable to reach such objectives, as exposed by Pr. Walinjom Muna from NCDH [8]. However, providing any IT solution for developing countries has to face a specific set of requirements, mainly linked to the inherent lack of a reliable infrastructure, weakness in the commercial and industrial environment, and the limited financial resources of governments.

Two main approaches have been envisioned to provide data services (such as health related data services) in developing countries. The first one is to provide a global and reliable electronic health record infrastructure. It usually requires huge financial investments, transversal agreements between the legislator, government, commercial partners and health practitioners. Besides, it faces problems related to network connectivity and coverage, central system architecture, administration and maintenance costs, unified data models and norms, security procedure and trusted authorities, legal and ethical frameworks, authentication and unique identification numbers for patients, etc. Representative initiatives of this approach include Google "Loon for All" and Facebook "Internet.org" projects. The focus of these initiatives is on improving network connectivity, by means of high altitude balloons or drones that act as gateways to the Internet. These projects are amazing and aim at bridging inherent lacks of developing countries, but they face difficult technological issues (e.g., Google solar helium balloons can only fly for a few days) and merely address a single dimension of the problem (network connectivity), ignoring other issues that need to be overcome to support reliable data-oriented services. Building a complete EHR following this approach can thus be considered as long-term initiative, with uncertain outcomes. The second approach for providing data services in developing countries consists in using existing infrastructure, whatever weak, with the target to quickly and practically offer working solutions to specific problems. Representative of this approach are solutions based on the use of phones and text messages to address specific issues, like keeping track of vaccine cold chains [12] or reminding medical appointments to patients. Many mHealth applications are envisioned on such basis [7]. As another example, we can indicate the existence of applications that connect patients and health counselors to help improve habits that may reduce the impact of a given disease [13]. However, while the advantages may be achievable in the short term, such proposals usually focus on specific scenarios and cannot be generalized to a broader scope of data-driven applications.

Motivated by the aforementioned shortcomings, we try here to bridge the gap between the two approaches (at least partially). We have recently proposed the vision of Folk-IS (Folk-enabled Information System) [14, 15], a new generic information system suited to provide generic data-oriented services. This proposal matches three requirements identified with NGOs (non-governmental organizations), which must be met by any practical ICT solution: (1) **Self-sufficiency**: the solution must not rely on any quick improvement of the existing software and hardware infrastructure but should benefit from it; (2) **Low cost**: very low initial investment, deployment and maintenance costs are assumed, the usual scale being a few dollars per user; and (3) **Privacy-by-design**: security and privacy are major prerequisites due to the lack of a global IT security infrastructure, secured servers, trusted authorities, and legal frameworks, leading to a self-enforcement in IT of citizens' privacy principles. The system would thus be based on some hypotheses made in the second approach mentioned above (exploiting the existing infrastructure), but with a wider potential in terms of application scope.

Folk-IS builds upon the emergence of very low-cost and highly-secure devices, and uses people moves to transparently and opportunistically perform data management tasks, so that IT services are truly delivered by all the folks (thus the acronym Folk-IS). The system raises new research challenges exposed in [14, 15] linked to the support of various architectural settings and personal devices, diverse data models, and heterogeneous, open and distributed infrastructures. In this paper, we instantiate the Folk-IS vision to provide PHRs, and turn Folk-IS into reality in the context of diabetes follow-up programs. Our contributions concentrate on (i) the design of an architecture which satisfies the use case without assuming any preexisting communication infrastructure, with strong privacy guarantees for the patients and at very low and incremental deployment costs; and (ii) new algorithms to enable the computation of global statistics on populations of patients with strong privacy and security guarantees.

The rest of this paper is organized as follows. Section 2 presents the diabetes follow-up use case. Section 3 proposes an architecture for diabetes follow-up based on PHRs, using the principles proposed in the Folk-IS vision. Section 4 investigates global computations with strong privacy and security guarantees. Section 5 concludes the paper.

2 Diabetes Follow-up Use Case

In Cameroon, more than 30.000 diabetic patients are followed by the *Cameroonian National Center for Diabetes and Hypertension* (NCDH). The current follow-up is based on paper folders. A first folder (*hospital folder*) is kept by the hospital and contains a summary of the most important information about the patient: personal attributes, treatments, and the results of the main medical examinations. A second folder (*patient folder*) is kept by the patient and contains his/her complete medical history. The patient provides it to the practitioner at the time of a consultation.

In practice, practitioners fill in the patient folder, which is the most complete one, and summarize some information in the hospital folder. Doctors also conduct epidemiological studies, by reporting data from the patient folder into the hospital folder during the consultation, such that after a given time period (e.g., one month), these data can be aggregated. These statistical studies are crucial for a good diabetes follow-up, because many important variables for the treatments depend on the location and the population's habits.

Some problems reduce the efficiency of the follow-up. First, the hospital folder may not be available during the consultation. Indeed, the hospital folder is sometimes kept by another service of the hospital, or remains at the archives department when the consultation has not been anticipated. Second, patients may lose their folders, and no backup is available. Third, statistics are difficult to compute since only a small subset of practitioners are involved in the collection of the data to perform a given statistics. To improve the follow-up at a low cost, we propose to replace the second folder by an electronic PHR held on a personal and secure portable IT device. In this way, the practitioners would have an easy access to the synthesis (equivalent to the hospital folder) from the patient folder even when the hospital folder is not available, and this synthesis may be synchronized on a server available in the hospital services (e.g., cardiology, ophthalmology, etc.) or other hospitals the patient visits. Moreover, the medical history would be searchable and more easily browsed. To avoid potential loss of data, the medical folders would be (securely) backed up and recovered if needed. And finally, a protocol would enable a convenient automatic computation of global statistics on many more patients (typically, all the patients visiting one or several hospitals in a given time frame) without hurting the privacy and security level of the system.

3 A Secure and Low Cost Architecture for Diabetes Follow-up

Our target is to reach the above objectives without relying on an existing infrastructure. Therefore, we use here the Folk-IS paradigm [14], which builds upon personal and secure devices currently emerging under different names and shapes (e.g., smart USB keys, secure SD cards, smart tokens, etc.). We term these devices *Folk-nodes*. From a hardware point of view, a folk-node is assumed to provide: (1) enough stable storage to host the complete medical history of its holder, (2) enough computing resources to securely run a server managing the data and enforce access control rules locally, (3) a tamper-resistant smart card to hold secrets (e.g., cryptographic material, certificates, passwords), (4) a biometric sensor (e.g., fingerprint reader) to ease the authentication of users, and (5) input/output capabilities (e.g., USB connector, and/or a wireless communication module) to interact with external devices (e.g., computers, smart phones, tablets, etc.).

The token pictured in Fig. 1 (left) is called *PlugDB* (see https://project.inria.fr/plugdb/en/) and is representative of what a folk-node can be. Inria provides it in open hardware, such that any electronic manufacturer can build it. When ordered in small quantities, its cost is around 60€. This is of the order of the average cost of one month of treatment for diabetes patients in Cameroun [6]. To further reduce this cost without jeopardizing the functionalities and privacy level, we propose to share the most expensive hardware components by integrating them on *Docking stations*. Docking stations would thus be required wherever the patient's PHR must be used. Typically, they could be placed in the consulting rooms, in the hospital hall, and at the registration desk of the hospital or service. Technically, building docking stations is easy since they simply integrate a subset of the hardware components of a folk-node, a USB slot, and a power outlet.

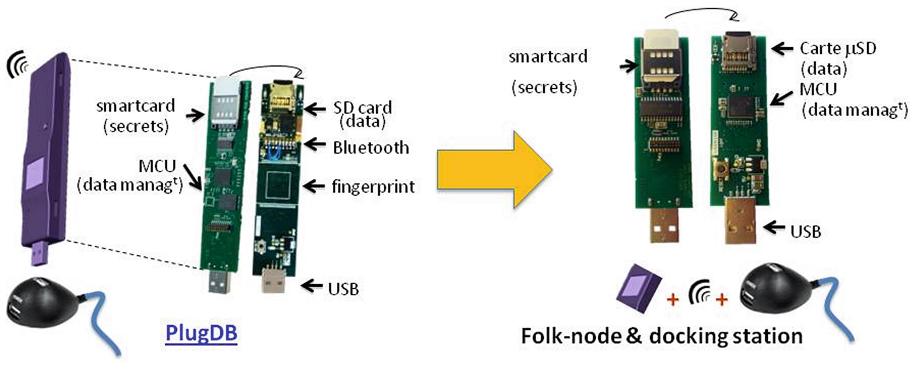

Fig. 1. Folk-nodes.

The wireless communication module and the fingerprint sensor are the two most expensive components of PlugDB. They can be integrated on shared docking stations as shown in Fig. 1 (right), thus reducing the price of the folk-node by an important factor (we estimate it at around 2 according to recent interactions with different manufacturers). Whenever a communication module is integrated on the docking station, the patients can plug their folk-node into the USB slot of the docking station to interact wirelessly with any practitioner device.

The fingerprint sensor is more difficult to share in the general case without reducing the security guarantees. Nevertheless, in a medical scenario, the system must only prevent folk-nodes from being used by somebody else than the patient owning it. To support these functionalities, the "fingerprint minutiae" of the patient and the code confronting the sensed fingerprint with these minutiae can be embedded in the patient's folk-node. The fingerprint sensor can thus be shared without lowering security, since it does neither store patients' minutiae nor the comparison algorithm resulting in unlocking the folder. Remark that we have also placed Flash memory sticks on the docking stations to be able to store (encrypted) data, e.g., intermediate results used while computing statistics, backup information, etc.

The overall architecture that we propose is pictured in Fig. 2. Each patient is equipped with a folk-node which contains his/her personal health record. Docking stations may be situated in the consultation rooms, at the registration desk and in the hospital hall. For simplicity, a local WiFi hotspot, which does not need to be connected to the Internet, is used to synchronize the data stored among the docking stations, to provide backup and global computations services (and potentially other distributed data services). Note that alternatively, opportunistic data exchanges could also be exploited (see [15] for details), but these techniques are not further considered here. Professionals can interact with the folk-nodes using a computer or their own personal device (smart phone or tablet), as advocated by BYOD (Bring Your Own Device) initiatives [10].

From the functional point of view, the patient first personalizes the folk-node at the first connection: a cryptographic public/private key pair is derived by the folk-node from a passphrase provided by the patient and his/her fingerprint is enrolled. At the time of the consultation, the patient unlocks her folk-node by using her finger, and the practitioner accesses the health record and may append new information. Any data inserted into the folk-node is also stored encrypted on the docking stations to generate an archive, and a synthesis view of this information can be synchronized on a hospital server if available. This will ensure the persistence of the data even if the folk-node experiences any problem. Thus, any lost folk-node could be restored from a blank (i.e., empty) folk-node, using the personal passphrase of the patient to regenerate the keys and access the backup information from any docking station. Similarly, patients who forgot their folk-node when coming to the hospital may borrow a blank folk-node at the registration desk, recover their folder from any docking station, and consult the practitioner using the borrowed folk-node. Any new data appended to the folder during the consultation will be integrated in the backup. The borrowed folk-node could then be reset after the consultation, since the new data will be integrated within the (forgotten) patient's folk-node at the next visit.

The backup and recovery mechanisms are not further described in this paper, but the reader can see [15] for information. Similarly, the embedded data management engine in charge of storing, indexing, querying and authenticating users on folk-nodes is out of the scope of this work; for more details, we refer the reader to two recent studies presenting an embedded relational database system [16] and a search engine designed for smart objects [17]. These proposals have been validated by a demonstration [18] and by an experiment in the field involving 40 patients and 80 medical-social professionals (see https://project.inria.fr/plugdb/category/medical/). The kernel of an embedded data management engine integrating these different proposals and a data recovery protocol based on the use of (encrypted) external storage will be provided as open source by Inria.

4 Distributed Computations

In this section, we investigate the problem of computing global statistics with strong privacy and security guarantees. We only concentrate on the computation of simple statistics (like average, sum, count, min, and max), and leave more complex computations for future work. We first introduce a basic algorithm without taking security into account. We then discuss techniques to enforce the security and privacy and propose an algorithm matching these requirements.

We describe the problem as follows. Some people are able to issue queries and are called *queriers* (e.g., practitioners, PhD students, researchers conducting an epidemiological study, etc.). They are equipped with personal devices (e.g., a smartphone or tablet). In several medical centers or hospital services, we deploy a set of docking stations. The set of docking stations deployed in the same hospital service can communicate synchronously using a WiFi spot. Each set of docking stations is called a *query spot*. Several *query spots* can be settled. We do not assume that all the queriers' personal devices and all the *query spots* can be accessible from the Internet, but we benefit from such Internet links when they exist. We propose to compute each query in three steps as follows:

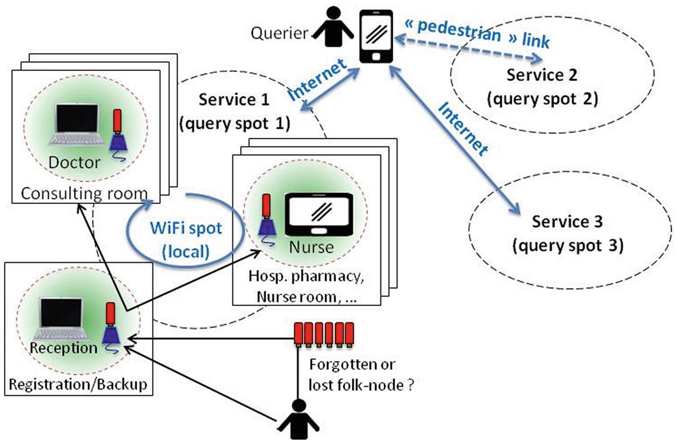

Fig. 2. Diabetes follow-up architecture.

1. First, the query is defined by a *querier*, and is transmitted to a single docking station of each *query spot*. The query can be either transmitted through the Internet if available, or by physically entering in the wireless communication scope of a given docking station of each of the *query spots* with the querier device using WiFi (or Bluetooth). The query is then automatically propagated to the other docking stations of the *query spot*.
2. Then, the folk-nodes connecting to a docking station in a given query spot will successively contribute to the running queries by enriching the intermediate results stored on the docking station. Docking stations act as transit points for intermediate results, processed and enriched successively by the participating folk-nodes. A given intermediate result on a docking station, when it aggregates enough contributions, is transmitted to all the docking stations of the query spot, and will not be modified anymore. By transmitting the results to all the docking stations instead of choosing only one, the querier does not need to connect to a specific docking station to retrieve the results.
3. Finally, the querier collects all the intermediate results available on each query spot by physically accessing a given docking station in each query spot, and merges together all the intermediate results to obtain the query result. If a sufficient number of contributions have been integrated into that result, the query can terminate, and all data related to that query are removed from the docking stations in the query spots.

The implementation of the first and third steps is simple and are thus not described in more detail. The second step deserves more explanation and a first (unsecure) implementation is given in Algorithm 1, considering the example of an average computation (other computations like sum, min and max can be deduced easily in a similar way). We assume in the algorithm that each query posted by a querier during step 1 generated an initial intermediate result that is set at *0* propagated to all the docking stations of the query spot. Then step 2 can be implemented as follows. When a folk-node connects to the docking station, the algorithm identifies the intermediate results the folk-node is eligible to contribute to, and updates the corresponding intermediate results by integrating its own contribution. Being "eligible" means that (the data of) the patient data does match the *scope of the query* (i.e., the specific query constraints that restrict which values will be considered to compute the required aggregation) and that the folk-node has not contributed to that query yet. When the folk-node contributes to an intermediate result, the number of contributions for that result is incremented by one. When the number of contributors reaches a given threshold, the folk-node transmits the intermediate result to all the other docking stations of the query spot.

Algorithm 1. Average query computation (launched when the folk-node connects to a docking station)

Input:
Q a set of tuples $\{ < id, id_q, s_q, v_q, n_q > \}$ describing the intermediate results of the queries, with id the identifier of the intermediate result, id_q a query identifier, s_q the scope of the query, v_q an intermediate result and n_q the number of contributors to this intermediate result;
D a set of personal values $\{d\}$ belonging to a certain patient;
P a set of query identifiers for which the folk-node has already contributed;
/* Note: *Q* is stored on the docking station, *D* and *P* are stored in the folk-node */
Output: $\emptyset$.

```
1.  while $\exists\, q \in Q, id_q \notin P$ do /* while the folk-node is eligible to a query q */
2.      $Q_r = \{ r \in Q, r.id_q = q.id_q, r.n_q < threshold \}$;
3.  /* Qr the set of tuples below the threshold for this query */
4.      $v = 0$; /* initialize the intermediate result value at 0 */
5.      $n = 0$; /* initialize the number of contrib. to the intermediate result at 0 */
6.      for each $r \in Q_r$ do /* for each interm. res. to be processed */
7.          $v = ( r.v_q * r.n_q + v * n ) / (r.n_q + n )$; /* aggr. the interm. res. values */
8.          $n = r.n_q + n$; /* increment the number of contributions */
9.          Delete the tuple with $id=r.id$ from the docking stations in the query spot;
10.     endfor;
11.     if $q.id_q \notin P, \exists\, d \in D, s_q(d) = true$ then /* if the folk-node must contribute */
12.         $v = v * n + d$; /* add the adequate contribution d to the intermediate result */
13.         $n$ ++; /* increment the number of contribution by 1 */
14.     endif;
15.     $id$ = generate_id( ); /* generate an id for the interm. result with the help of the docking station */
16.     Delete the tuples in $Q_r$ from the docking stations of the query spot;
17.     Insert $< id, q.id_q, q.s_q, v, n >$ on the docking station;
18.     $P = P \cup \{q.id_q\}$; /* add the query to the set of queries the folk-node has already contributed */
19. endwhile;
```

4.1 Ensuring Privacy and Security

The privacy and security of the patients' data must be guaranteed. First, to prevent from snooping intermediate results on the docking stations, which may lead to reveal individual contributions, we must encrypt the intermediate results. The tamper resistance smartcard embedded in each folk-node can be used to store the encryption key. A simple solution would be to distribute a shared secret key in each folk-node (stored in the tamper resistant smartcard and never exposed outside) and add an encryption/decryption operation in Algorithm 1 when the folk-nodes write/read intermediate results on/from the docking stations. However, although the smartcards exhibit a high level of tamper resistance [11], it is not possible in practice to totally prevent from any tampering against opponents that would be sufficiently motivated and equipped. In practice, if all the folk-nodes hold the same secret key, an attack targeting a single folk-node would potentially compromise all the successive intermediate results and all the individual contributions. To maximize privacy and security, two main conditions must be met: (condition 1) each folk-node must hold a unique private key, and (condition 2) the algorithm must lead each folk-node to be able to decrypt very few intermediate results. These two conditions are required to prevent from data snooping and ensure that attacking a single patient folk-node would only compromise a very small portion of the data, leading to a very low ratio between the cost of the attack (tampering into a smartcard is highly difficult and expensive) and its benefit (few intermediate results are revealed).

To satisfy the first condition, we consider that each folk-node holds a unique private/public key pair. To adapt our algorithms to the second condition, (i) we make the assumption that it is possible to know with a good probability the next folk-nodes which will connect to the docking stations in a query spot, and (ii) we design the algorithm in such a way that the querier cannot obtain intermediate results before a sufficient number of contributions (above a threshold) are aggregated, and (iii) we involve each contributing folk-nodes in the computations evenly, each one processing only a few intermediate results when connected to a docking station. Of course, the challenge is to adapt the algorithm without reducing the efficiency, i.e., the number of patients' folk-nodes which effectively contribute to a query within a given time frame. We show below that Algorithms 1 and 2 are comparable in terms efficiency (in number terms of the number of integrated contributions during a given time frame). A privacy-by-design and secure implementation of three steps of the computation algorithm presented above can be obtained as follows:

1. The querier generates a unique public/private key pair for each query, and posts the public part of the key on the docking stations along with the query.
2. Each folk-node is endowed with a unique public/private key pair, and provides its public key at the reception desk (one per query spot) which is inserted in an agenda and published on all the docking stations of the query spot. When a folk-node f_i connects to a docking station, among the intermediate results generated by previous folk-nodes f_j (different from f_i), a very small subset can be decrypted by f_i. If f_i can process intermediate results of a same query, it merges them into a single intermediate result, and if it is eligible to the query it integrates its own contribution into the intermediate result. If f_i is eligible to queries for which there is no intermediate result it can decrypt, it generates a new intermediate result containing only its own value. All the intermediate results decrypted by f_i are deleted from the docking stations of the query spot. All the intermediate results produced by f_i are encrypted with the public key of another waiting folk-node chosen in the agenda or with the encryption key of the querier if the number of contributions is above a threshold, and are send to all the docking stations in the query spot.
3. The querier collects the intermediate results produced in query spot by accessing a single docking station of the query spot (through the Internet, if available), decrypts them with its private key (only those results reaching a sufficient number of

contributions are accessible to the querier) and merges these intermediate results to obtain the query result. If a sufficient number of contributions have been integrated into the result, the query can be terminated, and all information related to this query is removed from the docking stations.

We show in Algorithms 2 an implementation of the step 2, respecting the security conditions listed above. Step 1 and 3 are not further detailed, due to space constraints. Notice that only the value v_q of each tuple needs to be encrypted, as the rest of elements in the tuple are not considered to be sensitive.

We discuss here some details of Algorithm 2. First, note that if the agenda can be assumed as exactly reflecting the order in which the folk-nodes will connect to the docking stations, the intermediate result produced by a folk-node connected to a docking station could be encrypted with the public key of the longest waiting folk-node in the waiting room, which is the first one in the agenda (i.e., in line 16 of Algorithm 2, we could set $dest = 0$, leading to choose $A[0]$ to encrypt the intermediate results). But in practice, this hypothesis does not hold since the patients do not all wait the same time before their consultation, depending on the type of care they need (e.g., a patient waiting for an injection performed by a nurse may be treated before another patient waiting for its consultation with a doctor, although he/she was registered after by the reception desk, and thus appears after in the agenda). In that case, choosing $dest = 0$ would lead the patients waiting the more to process more intermediate results, leading to hurt the condition 2 expressed above. To solve the problem, and evenly distribute the processing of intermediate results by the folk-nodes, we introduce in line *16* of the algorithm some randomness in the choice of the public key in the agenda used to encrypt the produced results. A second remark is that there is no need to lock the tuples accessed by the folk-node (Q_r accessed in line *3* of Algorithm 2) because these tuples can only be updated by the unique folk-node owning the appropriate private key to decrypt them. Third, at the end of the day, some intermediate results integrating a number of contributions below the threshold may be lost since the agenda is empty (the patients coming the next day may not be known yet). This is however not considered as a big issue, since simple solutions can be proposed. For example, some time before the end of the day, we may stop releasing intermediate results to the querier when the threshold is reached, but instead continue to integrate new values in these results. As a consequence, the values produced at the end of the day would not be lost but integrated into existing intermediate results above the threshold, leading to lose fewer contributions.

Algorithm 2. Secure average query computation (lauched on the folk-nodes)

Input:

Q a set of tuples $\{<id, id_q, s_q, v_q, K, K_{querier}, n_q>\}$ describing intermediate results of queries, with *id* the identifier of the intermediate result, id_q the query identifier, s_q the scope of the query, v_q an encrypted intermediate result, *K* the public key of a folk-node used to encrypt the intermediate result v_q, $K_{querier}$ the public key of the querier and n_q the number of contributions integrated to this intermediate result;

D a set of personal values $\{d\}$ where *d* belongs to the scope of the query *q* if $s_q(d) = true$;

P a set of query identifiers for which the folk-node has already contributed;

$\boldsymbol{K_{pub}}$ and $\boldsymbol{K_{priv}}$ are the public and private keys of the folk-node;

nb_active_ds the number of active docking stations in the query spot;

A an agenda with the public keys $[K_{next}]^{\downarrow}$ of the folk-nodes waiting for a consultation, ordered by registration date (a[0] has the oldest date, attributed to the folk node waiting for the longest);

/* **Note: *Q* and *A* are stored on the docking station; *D*, *P*, K_{pub}, K_{priv}, *A* are stored in the folk-node** */

Output: ∅.

```
1.  Remove the entry containing $K_{pub}$ from the agenda $A$;
2.  while $\exists\, q \in Q,\ q.K = K_{pub}$ or $q.id_q \notin P$ do /* while the folk-node has to process intermediate results */
3.      $Q_r = \{ r \in Q,\ r.id_q = q.id_q,\ r.K = K_{pub} \}$; /* Qr the tuples to be processed */
4.      $v = 0$; /* initialize the intermediate result value at 0 */
5.      $n = 0$; /* initialize the nb of contributions to the intermediate result at 0 */
6.      for each $r \in Q_r$ do /* for each interm. res. to be processed */
7.          $v = (\mathrm{Decrypt}_{Kpriv}(r.v_q) * r.n_q + v * n) / (r.n_q + n)$; /* aggr. the intermediate result values */
8.          $n = r.n_q + n$; /* increment the number of contributions */
9.          Delete the tuple with $id = r.id$ from the docking stations in the query spot;
10.     endfor;
11.     if $q.id_q \notin P,\ \exists\, d \in D,\ s_q(d) = true$ then /* if the folk-node must contribute */
12.         $v = v * n + d$; /* add its contribution d to the intermediate result */
13.         $n$++; /* increment the number of contribution by 1 */
14.     endif;
15.     if $n < threshold$ then /* if the nb of contributions is below the threshold */
16.         $dest$ = rand_int($nb\_active\_ds$); /* choose at random an entry between 0 and nb_active_ds */
17.         $K_{dest} = A[dest]$; /* read in A the corresponding public key */
18.     else /* the number of contribution has reached the threshold */
19.         $K_{dest} = K_{querier}$; /* choose the public key of the querier */
20.     endif;
21.     $v = \mathrm{Encrypt}_{Kdest}(v)$; /* encrypt the interm. result with the public key $K_{dest}$ */
22.     $id$ = generate_id( ); /* generate an id for the interm. res. with the help of the docking station */
23.     Insert $< id, q.id_q, q.s_q, K_{dest}, q.K_{querier}, n >$ into the DS of the query spot;
24.     $P = P \cup \{q.id_q\}$; /* add the query to the set P of query the folk-nodes has contributed to */
25. endwhile;
```

4.2 Sample Scenario

To give a rough estimate of the validity of the algorithm, we have implemented a simple simulation set up with representative parameters form the Cameroonian National Center for Diabetes and Hypertension in Yaoundé. We consider a single query spot, with an average of *10* practitioners, nurses and medical assistants available to treat the patients, using *10* docking stations, an additional docking station used at the reception desk to generate the agenda, and around 500 patient's visits per day.

Figure 3 shows the number of contributions being aggregated into the intermediate results for a query and made accessible to the querier along the day. In the X-axis, we plot the number of patients connecting successively along the day (from 1 to 500), and in the Y-axis we give the number of contributions to the query. The reference curve (called *contributions*) gives the total number of contributions made by the patients (one

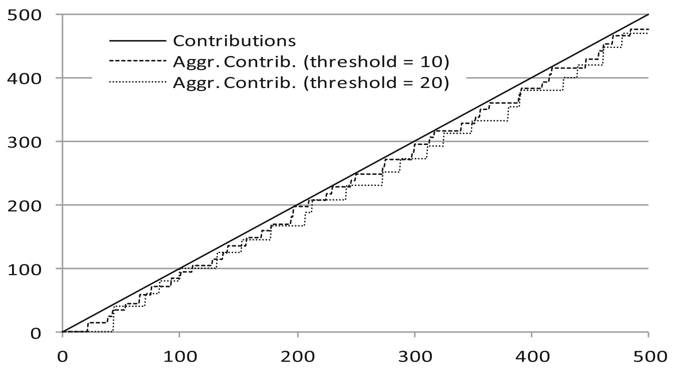

Fig. 3. Number of contributions available to the querier along time.

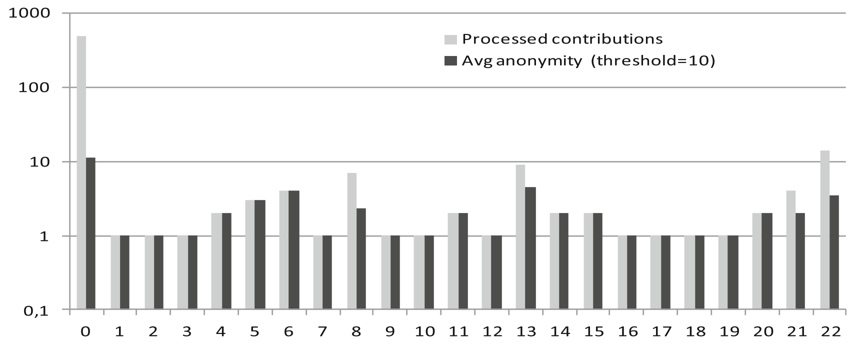

Fig. 4. Number of contributions processed by each participant and anonymity.

contribution per patient). The two dashed curves show the number of contributions integrated in the intermediate results released to the querier with a threshold at 10 and at 20. We conclude that a very small number of contributions escape from the intermediate results (limited to the contributions collected at the end of the day). Note that the third optimization proposed above could solve this problem.

Figure 4 shows the number of contributions that are exposed to the different actors taking part into the computation. In the X-axis, we plot the identifier of the querier (number *0*) and the folk-nodes of the patients successively connecting to the docking stations (for the sake of clarity, only the first 25 interactions are represented, numbered from *1* to *25*, but the data is representative for the next interactions). In the Y-axis, we plot the number of processed contributions (clear grey bar) and their level of anonymity (dark grey bar). A certain number of contributions has been integrated in the intermediate results each folk-node process, and we use the average number of contributions already aggregated in the intermediate result as an anonymity value (the higher the number of aggregated values, the more anonymous the value of each contributor). For example, folk-node *13* aggregates the value of two intermediate results which was obtained by aggregating 8 values, leading to an average anonymity of *4*. Folk-node *22* aggregates four intermediate results made of *14* values thus having an average anonymity of *3.5* (note that the produced intermediate result can be released to the querier, as the threshold is set at *10* in this scenario). At the end of the day, the querier (number *0*) has access to *476* contributions, aggregated into *42* intermediate results with at least a level of anonymity of *10* (the value of the threshold). The figure also shows that the algorithm evenly distributes the processing over all the participating patients' folk-nodes.

We conclude from this section that Algorithm 2 prefigures a good practical solution, with a high rate of contributions accessible to the querier and an implementation which exhibits good privacy and security "by-design" since the three conditions introduced above are well respected.

5 Conclusion and Future Work

This paper proposes a concrete architecture for managing personal health records with a potentially weak infrastructure, while fulfilling three main requirements: self-sufficiency, low cost and privacy-by-design. It builds upon the emergence of low-cost personal and secure devices, so called folk-nodes, coupled with docking stations, to manage personal health records. With this architecture, practitioners will be able to interact with the patient's record at the time of the consultation from a regular computer, smartphone or tablet. We provide in this paper the bases of a new secure distributed protocol to compute global statistics, without hurting the privacy and security level of the system. The main difference between this protocol and existing approaches is that it does not require any central server to be managed and provides a very high level of security. If a secure folk-node is broken (i.e., its cryptographic keys are released), very few amounts of data and intermediate results are compromised. As future work, we plan to precisely evaluate the accuracy and level of privacy of the protocol, and adapt it to support more complex statistics.

Acknowledgement. This work was partially done in the CICYT Project TIN2013-46238-C4-4-R and DGA-FSE.

References

1. Global status report on noncommunicable diseases 2014. Geneva, World Health Organization (2012)
2. International Diabetes Federation. IDF Diabetes Atlas update poster, 6th ed. Brussels (2014)
3. Global Health Estimates: Deaths by Cause, Age, Sex and Country, 2000-2012. Geneva, World Health Organization (2014)
4. Mathers, C.D., Loncar, D.: Projections of global mortality and burden of disease from 2002 to 2030. PLoS Med **3**(11), e442 (2006)
5. Liu, L., Yin, X., Morrissey, S.: Global variability in diabetes mellitus and its association with body weight and primary healthcare support in 49 low and middle-income countries. Diabet. Med. **29**(8), 995–1002 (2012)

6. Ngassam, E., Nguewa, J.-L., Ongnessek, S., Foutko, A., Mendane, F., Balla, V., Limen, S., Orr-Walker, B., Sobogwi, E., Mbanya, J.-C.: P318. Coût de la prise en charge du diabète de type 2 a l'Hopital central de yaounde. Diabet. Metab. **38**, A105 (2012)
7. Aranda-Jan, C.B., Mohutsiwa-Dibe, N., Loukanova, S.: Systematic review on what works, what does not work and why of implementation of mobile health projects in Africa. BMC Public Health **14**(1), 188 (2014)
8. Muna, W.F.: Comprehensive strategies for the prevention and control of diabetes and cardiovascular diseases in Africa: future directions. Prog. Cardiovasc. Dis. **56**(3), 363–366 (2013)
9. Kyazze, M., Wesson, J., Naude, K.: The design and implementation of a ubiquitous personal health record system for South Africa. Stud. Health Technol. Inform. **206**, 29 (2014)
10. Morrow, B.: BYOD security challenges: control and protect your most sensitive data. Netw. Secur. **2012**(12), 5–8 (2012)
11. Ravi, S., Raghunathan, A., Chakradhar, S.: Tamper resistance mechanisms for secure embedded systems. In: VLSI Design (2004)
12. Chaudhri, R., Borriello, G., Anderson, R.J.: Monitoring vaccine cold chains in developing countries. IEEE PerCom **11**(3), 26–33 (2012)
13. Mamykina, L., Mynatt, E., Davidson, P., Greenblatt, D.: MAHI: investigation of social scaffolding for reflective thinking in diabetes management. In: ACM SIGCHI (CHI 2008), pp. 477–486 (2008)
14. Anciaux, N., Bouganim, L., Delot, T., Ilarri, S., Kloul, L., Mitton, N., Pucheral, P.: Folk-IS: opportunistic data services in least developed countries. PVLDB **7**(5), 425–428 (2014)
15. Anciaux, N., Bouganim, L., Delot, T., Ilarri, S., Kloul, L., Mitton, N., Pucheral, P.: Opportunistic data services in least developed countries: benefits, challenges and feasibility issues. SIGMOD Rec. **43**(1), 52–63 (2014)
16. Anciaux, N., Bouganim, L., Pucheral, P., Guo, Y., Le Folgoc, L., Yin, S.: MILo-DB: a personal, secure and portable database machine. Distrib. Parallel Databases **32**(1), 37–63 (2014)
17. Anciaux, N., Lallali, S., Popa, I.S., Pucheral, P.: A scalable search engine for mass storage smart objects. PVLDB **8**(9), 1–13 (2015)
18. Lallali, S., Anciaux, N., Popa, I.S., Pucheral, P.: A secure search engine for the personal cloud. In: ACM SIGMOD, pp. 1445–1450 (2015)