DACAR Platform for eHealth Services Cloud

L. Fan, W. Buchanan, C. Thümmler,
O. Lo, A. Khedim, O. Uthmani, A. Lawson
Faculty of Engineering, Computing & Creative Industries
Edinburgh Napier University, Edinburgh, UK
L.Fan@napier.ac.uk

D. Bell
Faculty of Medicine
Imperial College London
London, UK
D.Bell@imperial.ac.uk

***Abstract*—The use of digital technologies in providing health care services is collectively known as eHealth. Considerable progress has been made in the development of eHealth services, but concerns over service integration, large scale deployment, and security, integrity and confidentiality of sensitive medical data still need to be addressed. This paper presents a solution proposed by the Data Capture and Auto Identification Reference (DACAR) project to overcoming these challenges. The DACAR platform uses a Single Point of Contact, a rule based information sharing policy syntax and data buckets hosted by a scalable and cost-effective Cloud infrastructure, to allow the secure capture, storage and consumption of sensitive health care data.**

Currently, a prototype of the DACAR platform has been implemented. To assess the viability and performance of the platform, a demonstration application, namely the Early Warning Score, has been developed and deployed within a private Cloud infrastructure at Edinburgh Napier University. Simulated experimental results show that the end-to-end communication latency of 97.8% of application messages were below 100ms. Hence, the DACAR platform is efficient enough to support the development and integration of time critical eHealth services. A more comprehensive evaluation of the DACAR platform in a real life medical environment is under development at Chelsea & Westminster Hospital in London.

***Index Terms*—eHealth, Cloud, Platform as a Service, Security, Single Point of Contact, Information Sharing Policy, Data Bucket**

I. Introduction

The use of modern communication infrastructures in medicine and the ubiquitous provision of health care services are in general subsumed under the term "eHealth" [1]. It necessitates a total reform and digitalisation of a health care system, including its production, supply and management [2]. Such technical innovations are expected to improve the quality of health care services, while lowering both the capital and operational costs significantly. Hence, the governments of the USA [3], Canada [4], UK [5], Japan [7], Korea [2] and of the European Union [6] are keen to shift their traditional health care services to a new paradigm, and to make eHealth a top priority on their policy agendas.

Though considerable research effort has been made in the literature relating to developing individual eHealth services, there is a lack of an open eHealth services platform which would allow the integration of such services into flexible, reliable, multifunctional and cost-effective eHealth systems, as well as their large scale deployment and delivery. Furthermore, a key challenge in eHealth is to use captured patient data in multiple forms and contexts, while maintaining strict access rights. It has been pointed out that health care data are subject to a variety of threats and attacks, and that inconsistency and loss of data have resulted in severe consequences [10]. Therefore, a complete eHealth services platform should also provide mechanisms to reinforce the integrity, security, confidentiality and auditability of sensitive medical data throughout their life cycle [8].

The aim of the Data Capture and Auto Identification Reference (DACAR) project is to develop, implement, validate and disseminate a novel, secure, "in-the-cloud" service platform for capture, storage and consumption of data within a health care domain. The objectives of the project include:

- Development of novel distributed and secure infrastructures based on role and inter-domain security polices;
- Smart device and system integration platform based on novel digital forensic security technology;
- Generic risk assessment strategy for smart device and system integration;
- Clinical evaluation, dissemination & commercialisation.

The remainder of this paper is organised as follows. Firstly an overview of the DACAR platform is given in Section II, and then the design of the system components are discussed in Section III, including medical data capture (Section III-A), storage (Section III-B), and consumption (Section III-C). The current implementation of the DACAR platform together with its demonstration applications are outlined in Section IV, followed by preliminary evaluation results. Work related to DACAR is highlighted in Section V. Finally, conclusions and future work are presented in Section VI.

II. System Overview

A. System Model

DACAR's system model consists of the following concepts:

1) *Domain:* A domain refers to a distinct business area that is administered by a single organisation. A health care application may involve multiple domains, such as hospitals, pharmacies, insurance companies and research institutions. The domains cooperate with each other to form a Circle of Trust (CoT), where each CoT member keeps a registry of trustworthy services provided by other CoT members.

2) *User:* A user refers to a consumer of an eHealth application, which can be a person or an impersonated service. A

user must be a member of at least one domain, which is able to resolve the user's identity into a specific role.

3) *Object:* An object refers to any entity that is managed by an eHealth system, such as patients and medical devices. An object is identified by a unique identifier (UID) assigned by its owner domain. An eHealth system should withstand content oriented and contextual privacy attacks, which means that even if an adversary has the capability of disclosing sensitive information from storage or communication channels, the adversary is neither able to find out that the information is associated with which object, nor to link the actual source and the destination of a message [11]. Therefore, opaque object pseudonyms shall be used in place of object UIDs [12].

4) *Attribute:* An object is described by a set of attributes, which are atomic units of information of primitive data types. For example, a patient object may comprise of a name attribute of string type, and of blood pressure and heart rate attributes of float type. Keeping attributes atomic offers two advantages. Firstly, it is convenient to generate complex medical documents, e.g. electronic health records, from atomic attributes dynamically. Secondly, it is also flexible to share atomic attributes among domains under the governance of fine-grained information sharing polices.

In practice, not only the core value of an attribute needs to be stored, but also a number of relevant meta data, e.g. the unit, capturer, location, time and device used to collect the data. When sufficient meta data are preserved, an eHealth application is able to document medical events occurring in the past, and to reconstruct them accurately at a later time.

5) *Service:* DACAR adopts a Service Oriented Architecture (SOA) to support the integration of eHealth services for data capture, storage and consumption purposes. A service refers to a course-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled, message-based communication model [13]. From a technical point of view, SOA captures many of the best practices of previous software architectures, including abstraction, autonomy, testability, loose coupling, reusability and statelessness.

6) *Hosting Infrastructure:* DACAR considers Cloud computing environment as its primary hosting infrastructure. The term "Cloud computing" became popular in 2007, with more than 20 definitions given in [14]. The characteristics of Cloud computing appear well-suited to meet the demand of eHealth applications, because a Cloud is inherently service oriented, loose coupling and strong fault tolerant [15]. Also, the business model of Cloud computing can significantly reduce the IT expertise and financial resources for small and medium sized participants to embark on large scale eHealth applications.

From a computing resource provision point of view, Cloud systems are broadly divided into three categories: public Clouds, private Clouds and hybrid Clouds [16], [17]. Public Clouds are confronted with more security challenges, and thus it is more difficult to guarantee the security of data stored in a public Cloud [18]. Hence, DACAR uses a private Cloud for data storage, and considers a hybrid Cloud for hosting

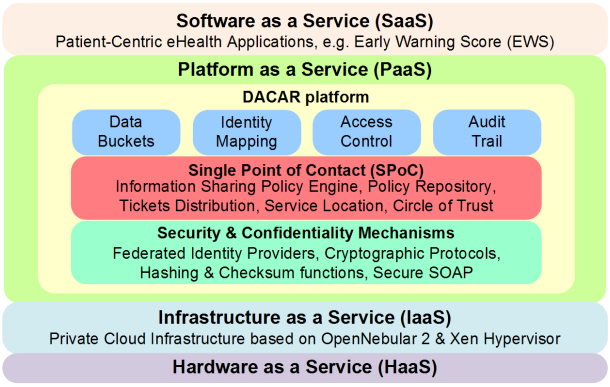

Fig. 1. Conceptual structure of the DACAR platform

service instances. From a service provision point of view, Cloud systems can be classified into at least four categories: Hardware as a Service (HaaS), Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [14]. DACAR focuses on the PaaS layer, and aims at providing developers with a platform that addresses the most common requirements of eHealth applications.

B. DACAR PaaS

The DACAR PaaS provides a set of foundation services, so that developers do not need to implement eHealth applications from scratch and deal with common issues repeatedly. Typical requirements of eHealth applications include:

- Authentication – cryptographic protocols that allow an entity to prove to a remote end its identity.
- Authorisation – individual- and role-based policies that endow entities with access rights to resources.
- Data Persistence – long-term storage of medical attributes, including their core values and meta data.
- Data Integrity – functions that ensure data are accurate, complete and consistent during any operations.
- Data Confidentiality – mechanisms that assure stored or transmitted data are accessible only to those authorised to have access, yet well protected from possible disclosure.
- Audit Trail – mechanisms that keep track of a chronological sequence of audit records pertaining to internal and external events and their implications.

Figure 1 shows a three-layer architecture of DACAR PaaS. At the bottom are **Security and Confidentiality Mechanisms**, which are used to meet the authentication, data integrity and confidentiality requirements. DACAR supports federated identity providers running a range of user authentication protocols, from traditional RADIUS [19] and Kerberos [20], to recent OpenID [21] and U-Prove [22]. In addition, the DACAR platform provides libraries and APIs for application developers to implement secure SOAP [23] services. This allows a number of security functions, e.g. digital signature, integrity checksum, hashing and encryption to be applied to application-specific portion of communication payload.

In the middle is the **Single Point of Contact** (SPoC), which is used to meet the authorisation requirement. A SPoC consists of two parts: a policy repository and a policy engine. The policy repository holds domain ontology, i.e. definitions of identities, roles, operations, services, objects, attributes and access rights. Each SPoC represents a single domain, and multiple SPoCs form a peer-to-peer (P2P) network that represents a CoT. Information requests are routed through the P2P network to an appropriate SPoC, which uses its policy engine to check the requester's identity, role, and grants access rights according to existing rules in the policy repository. A SPoC authorisation is issued in the form of a *Service Ticket* or a *Data Ticket*, which are security tokens protected by the SPoC's digital signature.

On the top are four system services:

1) The **Data Bucket** service offers long-term persistence of attributes and supports the Creation, Reading, Updating and Deletion (CRUD) of attribute values and associated meta data. Each attribute is stored in a single data bucket hosted by a Cloud infrastructure, and its CRUD service endpoint is registered with the SPoC of the attribute owner domain.

Any application service can put/get data to/from a data bucket, as long as it satisfies two conditions. Firstly, the service needs to know the qualified name of the target attribute, which is defined by domain ontology. Secondly, a rule needs to be established in the SPoC's policy repository to allow the service, or in the case of impersonation, the service invoker's identity or role, to perform CRUD operations over that attribute. If both conditions are met, the service is able to make a data request to the SPoC, which replies with a Data Ticket, carrying a reference to the CRUD service endpoint, a list of authorised operations, period of validity, and one-off session keys encrypted by the public keys of the requester and the CRUD service respectively. A Data Ticket may also carry data anonymisation and sanitisation instructions for the CRUD service to follow, as required by security policies.

2) The **Identity Mapping** service resolves user and object identifiers into pseudonyms, and vice versa. To enhance the contextual privacy of an eHealth application, opaque pseudonyms should be used in place of transparent user and object Ids, e.g. $12478c1abd$ instead of $PatientNo.253$. Hence, the DACAR platform uses pseudonyms whenever it is possible, and only reveals real identities to authorised individuals, roles and services when it is absolutely needed.

3) The **Access Control** service enables patients to create, edit and remove information sharing policies about their own attributes. DACAR adopts a patient-centric point of view, and regards a patient as the real owner of his/her medical data. Hence, the rights of access to such data should be defined by the patients themselves to their trust circle. The access control service provides a friendly user interface, so that authenticated users can easily set up policies controlling what personal information is available to whom, and what medical services they would like to subscribe to.

4) The **Audit Trail** service gathers text-based logs from application services, showing who was the active user, and what operations the user has performed during a given period of time. eHealth applications can benefit from the audit trail service in many different ways. Firstly, a sufficiently detailed audit trail enables the reconstruction of medical events and scenarios. Secondly, it keeps track of changes made to a system, and helps to roll them back when necessary. Thirdly, it provides evidence for digital forensics technologies, such as the Digital-DNA [25], to detect security anormalies and carry out countermeasures automatically. Finally, it facilitates the monitoring and analysis of the usage of computing resources, and thus helps to improve on the scheduling and load-balancing of the underlying Cloud infrastructure.

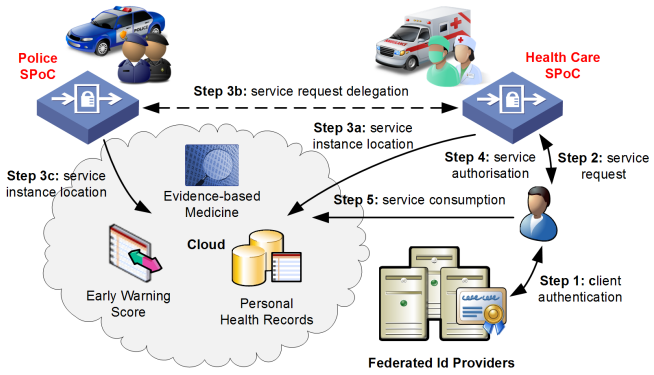

Fig. 2. DACAR application service consumption process

C. Work Flow

Figure 2 shows a storyboard of DACAR's work flow. Typically, a user consumes an eHealth service developed on the DACAR platform, in five steps as below:

Step 1 *Authentication:* The user logs on from one of the federated identity providers using a user name and a password, or other unique personal biometric information.

Step 2 *Request for a service:* The user's client software forwards the security credential obtained in step 1 to a responsible SPoC, together with a service request.

Step 3 *Instantiate the service:* The SPoC checks the user's identity, resolves it into a role, and matches the service request to existing security policies. In the case that the service is provided by the local domain, the SPoC is able to tell whether the user is allowed to consume this service, and to locate the service endpoint within the Cloud. However, if the service is provided by a foreign domain in the CoT, the SPoC will route the service request to another SPoC over the P2P network. For example, when a clinician needs to make contact with a patient's relatives in an emergency, he sends a request for a police registry service to the local health care SPoC, which forwards the request to a remote police SPoC.

Step 4 *Authorisation:* If the service request is permitted by corresponding security policies, the SPoC that made the decision creates and signs a Service Ticket. This contains the user's pseudonym and role (supplied by the user's local SPoC),

Fig. 3. Smart medical handheld devices

Table	Column	Data Type	Constraint
Core Data	Id	Integer	Primary Key
	MetaId	Integer	Foreign Key
	Value	String	Not Null
Meta Data	Id	Integer	Primary Key
	Unit	String	Not Null
	Object	Guid	Not Null
	Capturer	Guid	
	Device	Guid	
	Location	Guid	
	Time	DateTime	Not Null

TABLE I
DESIGN OF THE DATA BUCKET SCHEMA

a reference to the service endpoint, period of validity, and one-off session keys that enable the user's client software and a service instance to establish a secure SOAP session. Otherwise, a message is returned to tell the reason for rejection.

Step 5 *Consume the service:* Finally, the user's client software initiates a secure session using the information provided in the Service Ticket and starts to consume the service. If the service requires CRUD operations over certain attributes, the service itself becomes a consumer of related Data Bucket services. In this case, the service needs to go through Step 1 to 4 to obtain necessary Data Tickets from a SPoC using the service's own identity, or the service consumer's identity and role. In the latter circumstance the service is "impersonated", and shall use the Service Ticket received from its consumer as a complementary security credential – a Service Ticket carries a user's pseudonym and role, and is signed by a SPoC authority, as explained in Step 4.

III. DESIGN

The main objective of the DACAR project is to support the development and integration of eHealth services for the *capture*, *storage* and *consumption* of sensitive medical data. This section elaborates on DACAR's system components and approaches in accordance with these three aspects.

A. Data Capture

DACAR's approach for simple, efficient and secure capture of medical data involves four components: Radio Frequency Identification (RFID), smart mobile devices, hybrid network connectivity and secure SOAP services.

Radio Frequency Identification is a technology that can be used to identify, authenticate, track and trace medical objects, as well as to gather information about them and their environment [26]. An RFID system consists of a transponder tag, a reader, a software programme for processing the data collected, and a database for data persistence. RFID transponder tags can be *passive* or *active*. Passive tags have no processing capability and no internal power source, and thus can only work within a short range. DACAR employs passive RFID tags for identification purposes, e.g. patient wristbands. In contrast, active RFID tags have processing and storage facilities supported by an internal power source, so they can work like microcomputers and transmit data up to tens of meters. Active RFID tags can be integrated with sensors, which raise awareness about a medical context. DACAR uses active RFID tags to collect patients' medical data in real time.

Smart mobile devices, such as mobile phones, PDAs and tablet PCs are used as RFID readers, which collect attribute values from RFID tags, and then transmit them to corresponding Data Buckets in the Cloud. Figure 3 illustrates the PDAs and tablet PCs designed and manufactured by CipherLab UK for the DACAR project. These devices are based on Windows or Windows Mobile operating systems, and feature easy to decontaminate coating, video camera, bar code reader, RFID reader, bluetooth and WiFi modules. Also, fingerprint access control guarantees that the devices can only be used by authorised medical staff.

A RFID reader requires a communication channel to relay the captured data to the application layer. DACAR uses **hybrid network connectivity**, which comprises of a wired Intranet backbone, fixed wireless access points, roaming mesh routers, and cellular network. The wired Intranet can cover the main buildings of a health care institution, connecting workstations in offices and wards to web and database servers. This network can be extended by a number of fixed wireless access points to cover areas in the vicinity of the main buildings. The range of wireless signal can be further extended on demand by roaming mesh routers [27] when a temporary ad-hoc network is needed in emergencies. In the case of RFID reader based-on a smart mobile phone, it would be possible to access the 3G/4G cellular network as well. This is especially helpful for medical staff on the move, e.g. sending patient data to a hospital from an ambulance for rapid diagnosis purposes.

A major concern for transmitting confidential medical data over a wireless or public network is security. To address this, the DACAR PaaS provides libraries and APIs for implementing **secure SOAP services**, as discussed in Section II-B. A SPoC is used to distribute Service and Data Tickets, so that application participants are able to follow the same protocol to encrypt and decrypt their messages.

B. Data Storage

After a considerable amount of medical data are captured, a scalable solution is needed for storing them. The DACAR platform keeps attributes in an atomic format to enhance their reusability and manageability, as discussed in Section II-A. A third important rationale behind this design is that an atomic attribute can be physically hosted by a single data bucket, which is convenient to deploy, migrate, redeploy and back-up

Query Syntax	Description
Identifiers	Column names, e.g. Value, Unit and Object
Literals	Literal values, e.g. "Alice" and 3.14
Types	All C# primitive data types, Guid and DateTime
Operators	Arithmetic, e.g. $+$, $-$, $*$, $/$ and $\%$ Relational, e.g. $==$, $!=$, $>$, $>=$, $<$ and $<=$ Logical, e.g. &&, || and !
Variables	Transparent Ids, e.g. $Chelwest.CP.JD0$ Aggregate values, e.g. MIN, $MEAN$ and MAX Sequence numbers, e.g. SEQ, $FIRST$ and $LAST$
Predicates	Chain of queries, e.g. $[pre1][pre2][pre3]$...

TABLE II
SUMMARY OF DACAR'S DATA BUCKET QUERY SYNTAX

within a Cloud. Compared to a traditional database, the data bucket approach is designed to be more flexible and scalable.

A key to understanding DACAR's data buckets is their attribute-oriented character. For example, all patients' blood pressure data are uploaded to, and preserved by, the same data bucket that is dedicated to the Blood Pressure attribute. The way of distinguishing different data samples is through their meta data. Table I outlines the relational schema of a data bucket, where a core data entry only stores the value of an attribute, and a foreign key refers to a meta data entry that supplements the unit of that value, its owner object, and contextual information such as who captured the value, using which device, at what location and at what time.

I/O operations of a data bucket are carried out via the CRUD service, of which the service endpoint is registered with the SPoC that represents the owner domain of the corresponding attribute. Considering that the cost of storage capacity is becoming lower and lower, the DACAR platform deprecates conventional *Update* and *Delete* operations. Instead, it preserves all historical attribute values for audit trail purposes, and returns the most recent value of an attribute, where a single valid value of that attribute is required.

A challenge for the design of the CRUD service is efficient filtering of complex meta data. RESTful web services [24] are proposed to allow a service consumer to send dynamic queries in JSON [28] or AtomPub [29] format to a service provider, to retrieve exactly the data that the consumer needs. This approach can reduce the response time and the cost of network bandwidth and processing power for a data intensive web application. It is especially beneficial to mobile devices having limited computing resources and battery life. However, currently the DACAR platform mainly supports secure SOAP services, which is also the technology used to implement the CRUD service. Until libraries and APIs for secure RESTful services are completed, the DACAR platform currently provides an easy to use query syntax, as outlined by Table II, so that a service consumers can make a dynamic data request over SOAP, without requiring a service provider to implement large numbers of cumbersome web methods.

Firstly, the query syntax supports **Identifiers**, i.e. column names of the data bucket schema, **Literal** values, and all primitive data **Types** of the C# programming language, as well as the Global Unique Identifier (Guid) and DateTime composite data types. Secondly, it supports common arithmetic, relational and logical **Operators**. Thirdly, it supports **Variables**, which are names or key words between two dollar signs. A variable is needed in the following circumstances:

- Transparent IDs – it is likely that a service consumer only knows the transparent ID of a target object, whereas the object is referred to using an opaque pseudonym in security policies and data buckets. In this case, the service consumer can remind a CRUD service to resolve the ID into a pseudonym using the Identity Mapping Service, by putting the ID between dollar signs.
- Aggregate Values – these are similar to aggregate functions of SQL.
- Sequence Numbers – the CRUD service sorts data samples in an intermediate result set according to their time stamps, and endows each of them with a temporary sequence number, so that a service consumer can specify a sub set that is required from a large collection.

Finally, the query syntax supports a chain of **Predicates**. A predicate is a partial query between two square brackets, and a chain of predicates serves as multiple "Where" clauses in SQL to narrow down a result set. Also, a predicate provides the scope for a CRUD service to evaluate the actual value of a variable at run time. In other words, the intermediate result set is updated once a predicate is processed, and the actual value of an aggregate or sequence variable is changed accordingly.

It is easier to understand the query syntax using a concrete example. Suppose that an eHealth application needs to find out the last five body temperature samples, which were above $37.5\,^{\circ}\mathrm{C}$, of critically ill patient $JohnDoe$ at Chelsea & Westminster Hospital, since $9:30am$, $Jan\ 1^{st}$, 2011. The query below can be used in the case that the application only knows the patient's transparent Id as $Chelwest.CIP.JD0$:

```
[ Object == $Chelwest.CIP.JD0$ && Value >= 37.5 &&
  Time >= DateTime(2011,1,1,9,30,0) ][ $SEQ$ > $LAST$ - 5 ]
```

Another important issue that must be considered while designing the data bucket is data encryption. Here, a fundamental question is where the encryption should be performed. Application-level encryption means that sensitive medical data are encrypted and decrypted by application services, which is transparent to the underlying database engine. On the other hand, database-level encryption means that data are encrypted and decrypted by the database engine automatically before written to and read from the disk. Both approaches can protect data from storage attacks effectively, e.g. theft of storage media, but they both have distinct disadvantages. Application-level encryption suffers from considerable performance overheads, because the size of encrypted data can be much larger than the original plain text. Furthermore, the encryption process eliminates all data type information, as the original data have to be converted into strings. Comparatively, database-level encryption is more efficient and does not require any change to the application layer. However, it is vulnerable to malicious DBAs, which is an inherent problem.

Currently, the DACAR platform adopts database-level encryption, because sensitive medical data are only stored within a private Cloud, which is set up and maintained by trustworthy administrators. Oracle Transparent Database Encryption [30], or SQL Server Encryption [31] are used to encrypt the data buckets automatically. Of course application developers who plan to store data in a public Cloud can still resort to application-level encryption schemes such as [32] and [33].

C. Data Sharing

The most important goal of the DACAR platform is to allow trustworthy individuals, roles, application services and subsystems to consume captured medical data in many different ways, while maintaining strict access rights. This is achieved by DACAR's information sharing policy.

The policy syntax is designed as below:

[Permission] [Requester] [Operations] [Attributes] of **[Object]** with **[Context]** from **[Owner]** for **[Multiplicity]** records in **[Time Window]** using **[Compliance]**.

- **Permission** indicates the action of the rule and defines whether a request meeting the rule criteria will be *permitted* or *denied*.
- **Requester** identifies the source of a request as a specific individual or the membership of a certain role.
- **Operations** refer to *create*, *read*, *update* and *delete*. However, the DACAR platform deprecates *update* and *delete* operations, as discussed in the previous section.
- **Attribute** is a unit of information describing an **Object**.
- **Context** identifies the reason why the information is being shared. It also governs the level of access and permissions associated with information exchange, and thus affects the priority accorded to information requests.
- **Owner** defines a role with sufficient privileges to manage all aspects of an information element, and to permit or deny access to an information element, as required by legislation and defined responsibilities.
- **Multiplicity** defines the maximum number of records that can be shared over a period of time.
- **Time Window** defines the period of validity of a rule using ISO 8601 coordinated universal time format.
- **Compliance** refers to legislative requirements that affect the exchange of information, as well as data anonymisation and sanitisation instructions.

The policy elements discussed above can be used flexibly while composing security rules for different purposes:

1) *Service Authorisation:* A service authorisation rule allows or denies certain individuals or roles to consume an application service. In this circumstance, the **Object** element is used to identify a service that a rule is about, and **Attributes**, **Context**, **Multiplicity** and **Compliance** elements can be omitted.

2) *Service Subscription:* This represents a patient's subscription to a specific eHealth service, and thus automatically allows *creation* and *reading* of arbitrary numbers of attribute records, as needed by the service to fulfil its functionality. **Context** and **Multiplicity** elements can be omitted in this case.

Machine	Cloud Node A & B	Cloud Controller
Processor	Dual Core Xeon 3.0GHz $*$ 4	Quad Core Xeon 2.8GHz
Memory	32GB Dual Rank 800MHz	12GB DDR3 1,333MHz
Storage	73GB SCSI 10, 000RPM $*$ 6	1TB SATA 7,200RPM
LAN	Intel Pro 1000PT $*$ 4	Intel Pro 1000PT
WAN	1Mbps Optical Fibre	1Mbps Optical Fibre

TABLE IV
HARDWARE SPECIFICATIONS OF THE EXPERIMENTAL CLOUD

3) *Specific Consent:* This policy type enables a patient to grant access rights for his/her own attributes to trustworthy individuals and roles in a fine-grained manner. It also allows an impersonated service to access a patient's information, using its consumer's identity or role.

4) *General Consent:* Sometimes it is difficult for a patient to name the grantees for a specific consent, because they are unclear, unknown, or hard to describe. A general consent is useful in this situation to facilitate information sharing in a coarse-grained manner. The **Context** element is used for a patient to express the willingness to share his/her information with services for a certain purpose, from a certain domain, or above a certain level of importance.

5) *Investigation:* This policy type is only used in exceptional situations, such as a medical incident investigation, to obligate unconditional information sharing. Hence, elements such as **Requester**, **Context** and **Multiplicity** can be omitted, whereas the **Compliance** element may require the investigator to possess additional security tokens on a per case basis.

Table III gives a summary of DACAR's policy syntax.

IV. Implementation & Evaluation

Currently, a prototype of the DACAR platform has been implemented using C# and .Net 4.0. The SPoC is implemented as a self-hosting WCF service running in Windows Server 2008. A Windows Active Directory domain server is employed as the identity provider that authenticates users using the Kerberos protocol. Data Bucket, Access Control, Identity Mapping and Audit Trail services are all implemented as WCF services, which are hosted by IIS 7 web server. The back end of the Data Bucket is supported by SQL Server 2008, and the front end of the Access Control service is developed using WPF XAML, which can be run on Windows-based PCs and mobile devices. The policy engine is implemented in Java, and both information requests and rules are written in XML.

Furthermore, a proof-of-concept application, namely the Early Warning Score (EWS), has been implemented on top of the DACAR platform. EWS is a medical practice widely used in UK hospitals. The traditional EWS requires medical staff to record and enter six vital signs of a patient on a paper-based observation chart periodically, and to calculate a risk score according to predefined equations. In the case that a patient is evaluated to be "at risk", the medical staff should make contact with appropriate clinicians. The traditional EWS is prone to mistakes, as the measurement, recording and calculation work all need to be done manually. The new EWS application

Policy Elements	Service Authorisation	Service Subscription	Specific Consent	General Consent	Investigation
Permission	permit or deny	permit or deny	permit or deny	permit or deny	permit or deny
Requester	user pseudonym or role	service pseudonym	user pseudonym or role	n/a	n/a
Operations	read	create or read	create or read	read	read
Attributes	n/a	attribute pseudonyms	attribute pseudonyms	attribute pseudonyms	attribute pseudonyms
Object	service pseudonym	object pseudonym	object pseudonym	object pseudonym	object pseudonym
Context	n/a	n/a	n/a	domain & service level	n/a
Owner	domain pseudonym	domain pseudonym	domain pseudonym	domain pseudonym	domain pseudonym
Multiplicity	n/a	n/a	number of records	number of records	n/a
Time Window	period of validity	period of validity	period of validity	period of validity	period of validity
Compliance	n/a	sanitisation instructions	sanitisation instructions	sanitisation instructions	security token

TABLE III
SUMMARY OF DACAR'S INFORMATION SHARING POLICY SYNTAX

fully automates this process by capturing vital signs using RFID sensors, transmitting the values to data buckets using smart handheld devices, monitoring patient status constantly in real time, and notifying clinicians by calling or sending messages to their mobile phones. A similar application has been proposed in [9].

Currently, medical evaluation of EWS is under development at the Chelsea & Westminster Hospital in London. In the mean time, a private Cloud infrastructure has been set up at the Edinburgh Napier University to obtain preliminary experimental results. This Cloud consists of three physical machines, i.e. one Cloud controller running OpenNebular2 and two worker nodes running Xen hypervisor on Ubuntu [34]. The hardware specifications of the machines are given in Table IV. Such a configuration is sufficient to host four virtual machine instances for a SPoC server, a Data Bucket server, and two web servers for system and application services respectively. Software has been developed to simulate a number of virtual patients, uploading their vital signs to the data buckets.

Figure 4 shows the end-to-end latency distribution of 1000 application messages. 97.8% of the values were below 100ms, and the rest were between 100ms and 200ms. These results suggest that the DACAR platform imposes only a small delay on application-level messages, and thus is efficient enough to support the development and integration of time critical eHealth services. Although the latency in a real medical set-up might be higher than the simulated result, it should be within the millisecond level, which is still acceptable.

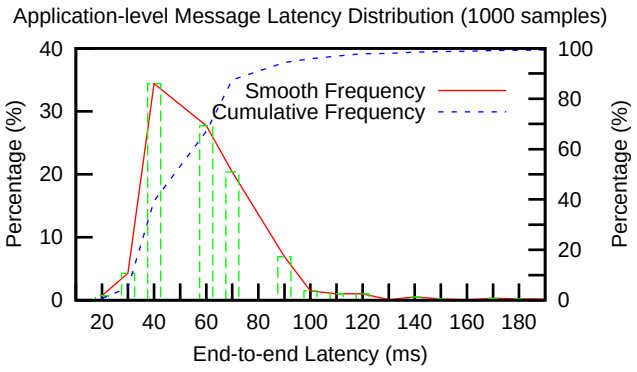

Fig. 4. Application-level message latency distribution analysis

V. RELATED WORK

Zhang et. al. have identified a set of security requirements for eHealth application Clouds and proposed a novel security model in [8]. This model is mainly designed for the sharing of Electronic Health Records (EHR), while the DACAR platform aims to support the development, integration and large scale deployment of a wider range of eHealth services.

Kilic et. al. have proposed to share EHRs among multiple eHealth communities over a peer-to-peer network [35]. A super-peer is used to represent an eHealth community, which is responsible for routing messages and adapting different meta data vocabularies used by different communities. This super-peer design is similar to a SPoC of the DACAR platform, yet a SPoC provides more authorisation functionalities.

For a patient-centric eHealth platform it is crucial to obtain various patients' consent in an electronic way. Coiera et. al. have identified four levels of e-consent, including *general consent*, *general consent with specific exclusions*, *general denial* and *general denial with specific consents* [36]. DACAR's policy syntax is able to express all of the above, as well as *service authorisation*, *service subscription* and *investigation*. Furthermore, Pruski has identified the requirements for an e-consent language to capture *specific grantees*, *operations*, *purposes* and *period of validity*, and proposed a novel language called e-CRL [37]. DACAR's policy syntax is as competent as e-CRL, and has been successfully applied to other domains besides health care, such as police and social care [38], [39].

Currently, DACAR does not support legacy medical records that are not based on atomic attributes. Amato et. al. have proposed a semantic based methodology to extract and classify atomic units of information from legacy "monolithic" medical documents [40]. It provides a useful complement to address this limitation of the DACAR platform.

VI. CONCLUSION & FUTURE WORK

This paper presents a novel eHealth services platform designed by the Data Capture and Auto Identification Reference (DACAR) project. Firstly, the DACAR platform facilitates the development of eHealth applications by addressing the most typical requirements, including authentication, authorisation,

secure data transmission, persistence, integrity, confidentiality and audit trail. Secondly, it provides a suite of hardware and software solutions to integrate the capture, storage and consumption of sensitive medical data. Thirdly, it supports large-scale deployment and delivery of eHealth services using a scalable and cost-effective Cloud infrastructure.

Key components of the DACAR platform include the Single Point of Contact (SPoC), the Information Sharing Policy, and the Data Buckets. This paper outlines the design and implementation of these components, as well as preliminary evaluation results obtained using a demonstration application called the Early Warning Score. The experimental results suggest that the DACAR platform imposes small communication latency on application-level messages, and hence it is sufficiently efficient to support the development and integration of time critical eHealth applications.

In future work, a comprehensive evaluation of the DACAR platform will be carried out in a real medical environment, and the design and implementation of the platform will be improved continuously. Another avenue of future work is to build bridges between DACAR and other commercial eHealth services platforms, such as Microsoft's Health Vault [41]. The goal is to enable secure sharing of health care information on a larger scale, and ultimately, to support expert-guided proactive patient-centric health care.

ACKNOWLEDGEMENT

This research is partially supported by grant from UK's Technical Strategy Board. Also, we acknowledge the hard work of the whole project consortium, including Edinburgh Napier University, Chelsea & Westminster NHS Foundation Trust, Imperial College London, Kodit Database Ltd, GS1 UK Ltd and CipherLab UK Ltd.

REFERENCES

[1] D. Slamanig and C. Stingl, "Privacy Aspects of eHealth," in *Proc. of ARES '08*. IEEE, March 2008, pp. 1226–1233.

[2] H. J. Cheong, N. Y. Shin, and Y. B. Joeng, "Improving Korean Service Delivery System in Health Care: Focusing on National E-health System," in *Proc. of eTELEMED '09*. IEEE, 2009, pp. 263–268.

[3] "Federal Health IT Initiatives," http://www.hhs.gov/healthit, June 2009.

[4] "Canada Health Infoway," http://www.infoway-inforoute.ca, June 2009.

[5] J. Dzenowagis and G. Kernen, "Global vision, local insight," World Health Organization Press, Report for the World Summit on the Information Society, 2005.

[6] "RIDE Deliverable 2.1.4 European Good Practices," http://www.srdc.metu.edu.tr/webpage/projects/ride/, 2010.

[7] "Japan: Emerging eHealth," *SCRIP Magazine*, pp. 21–24, March 2002.

[8] R. Zhang and L. Liu, "Security Models and Requirements for Healthcare Application Clouds," in *Proc of CLOUD '10*. IEEE, 2010, pp. 268–275.

[9] C. Rolim, F. Koch, C. Westphall, J. Werner, A. Fracalossi, G. Salvador, "A Cloud Computing Solution for Patient Data Collection in Health Care Institutions," in *Proc. of ETELEMED*. IEEE, 2010, pp. 95–99.

[10] M. Smith, W. Buchanan, C. Thuemmler, D. Bell and R. Hazelhoff, "Analysis of Information Governance and Patient Data Protection within Primary Health Care," To appear in Int. J. for Quality in Health Care.

[11] X. Lin, R. Lu, X. Shen, Y. Nemoto, and N. Kato, "Sage: a Strong Privacy-preserving Scheme Against Global Eavesdropping for eHealth Systems," *IEEE J-SAC*, vol. 27, no. 4, pp. 365–378, May 2009.

[12] T. Neubauer and A. Ekelhart, "An Evaluation of Technologies for the Pseudonymization of Medical Data," in *Proc. of SAC '09*. ACM, 2009.

[13] M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, M. Luo, and T. Newling, *Patterns: Service-Oriented Architecture and Web Services*. IBM Redbooks, July 2004.

[14] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, January 2009.

[15] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The Characteristics of Cloud Computing," in *Proc of SCC '10*. IEEE, 2010, pp. 275–279.

[16] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds," *Internet Computing*, vol. 13, no. 5, pp. 14–22, 2009.

[17] R. V. den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads," in *Proc. of CLOUD '10*. IEEE, 2010, pp. 228–235.

[18] B. Kandukuri, V. Paturi, and A. Rakshit, "Cloud Security Issues," in *Proc. of SCC '09*. IEEE, 2009, pp. 517–520.

[19] C. Rigney, S. Willens, A. Rubens, and W. Simpson, "Remote Authentication Dial In User Service," IETF standards RFC 2865, June 2000.

[20] B. Tung, *Kerberos: a Network Authentication System*. Addison-Wesley, May 1999.

[21] "OpenID Authentication 2.0," OpenID Foundation, December 2007.

[22] C. Paquin and G. Thompson, "U-Prove CTP White Paper," Microsoft, Tech. Rep., March 2010.

[23] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)," W3C Recommendation, April 2007.

[24] L. Richardson and S. Ruby, *RESTful Web Services*. O'Reilly, ISBN 0596529260, May 2007.

[25] J. Graves, B. Buchanan, and N. Bose, "DNA Digital Fingerprinting Framework," US Patent: 61/116681, UK Patent: 0816556.5, Nov 2009.

[26] C. Thuemmler, W. Buchanan, and A. Lawson, "Radio Frequency Identification (RFID) in Pervasive Healthcare," *Int. J. of Healthcare Technology and Management*, vol. 10, pp. 119–131, 2009.

[27] I. Akyildiz and W. Xudong, "A Survey on Wireless Mesh Networks," *Communications Magazine*, vol. 43, no. 9, pp. S23–S30, Sep 2005.

[28] D. Crockford, "The application/json Media Type for JavaScript Object Notation (JSON)," IETF standards RFC 4627, July 2006.

[29] J. Gregorio and B. de hOra, "The Atom Publishing Protocol," IETF standards RFC 5023, October 2007.

[30] M. Strole, "Oracle Transparent Data Encryption for SAP," Oracle White Paper, March 2010.

[31] S. Hsueh, "Database Encryption in SQL Server 2008 Enterprise Edition," Microsoft, SQL Server Technical Article, February 2008.

[32] H. Jumaa, P. Rubel, and J. Fayn, "An XML-based Framework for Automating Data Exchange in Healthcare," in *Proc. of Healthcom '10*. IEEE, 2010, pp. 264–269.

[33] Y. Ding and K. Klein, "Model-Driven Application-Level Encryption for the Privacy of E-health Data," in *Proc. of ARES '10*. IEEE, 2010, pp. 341–346.

[34] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Resource Leasing and the Art of Suspending Virtual Machines," in *Proc. of HiPC '09*. IEEE Computer Society, 2009, pp. 59–68.

[35] O. Kilic, A. Dogac, and M. Eichelberg, "Providing Interoperability of eHealth Communities Through Peer-to-Peer Networks," *IEEE TITB*, vol. 14, no. 3, pp. 846–853, 2010.

[36] E. Coiera and R. Clarke, "e-Consent: the Design and Implementation of Consumer Consent Mechanism in an Electronic Environment," *Journal of the American Medical Informatics Association*, vol. 11, no. 2, pp. 129–140, 2004.

[37] C. Pruski, "e-CRL: A Rule-Based Language for Expressing Patient Electronic Consent," in *Proc of ETELEMED*. IEEE, 2010, pp. 141–146.

[38] O. Uthmani, W. Buchanan, A. Lawson, and C. Thuemmler, "Novel Information Sharing Syntax for Data Sharing Between Police and Community Partners, Using Role-Based Security," in *Proc. of ECIW '10*. IEEE, 2010, pp. 394–402.

[39] B. Buchanan, L. Fan, A. Lawson, R. Scott, B. Schafer, C. Tuemmler, and O. Uthmani, "Interagency Data Exchange Protocols as Computational Data Protection Law," *Legal Knowledge and Information Systems*, vol. 223, pp. 243–147, December 2010.

[40] F. Amato, V. Casola, A. Mazzeo, and S. Romano, "A Semantic Based Methodology to Classify and Protect Sensitive Data in Medical Records," in *Proc. of IAS '10*. IEEE, 2010, pp. 240–246.

[41] "Microsoft HealthVault", www.healthvault.com.