Evaluating the Viability of Ciphertext-Policy Attribute-Based Encryption on Service-Oriented Health Information Exchange Systems

Edgardo Felizmenio Jr.
Computer Security Group
University of the Philippines - Diliman
Quezon City, Philippines
epfelizmenio@upd.edu.ph

Susan Festin
Computer Security Group
University of the Philippines - Diliman
Quezon City, Philippines
spfestin@dcs.upd.edu.ph

***Abstract*—Achieving interoperability in healthcare leads to several benefits that affect consumers and other stakeholders. With this, several countries have already implemented a Health Information Exchange (HIE) to enable interoperability in their areas. OpenHIE [5] is an initiative that aims to address the interoperability challenges in underserved regions, such as third-world countries. OpenHIE provides a framework for implementing an HIE which is based on the Service-Oriented Architecture (SOA). Its design has been proven to work, and other countries and organizations have shown interest in adopting its components. Currently, OpenHIE can provide the *de facto* standard for securing health information systems. However, several essential security features concerning the privacy of health information have yet to be implemented. Here we explore the application of Ciphertext-Policy Attribute Based Encryption (CP-ABE) on OpenHIE to realize the aforementioned security features. Results show that applying CP-ABE will not only change the implementation of the functionalities of the HIE, but it will also add an overhead to its non-functional characteristics, such as the number of transactions per second, memory, and bandwidth usage.**

***Index Terms*—Service Oriented Architecture, Health Information Exchange, Ciphertext-Policy Attribute Based Encryption**

I. Introduction

A health information system is an application for automating healthcare related tasks such as maintaining health records. Since these systems may store health-related information, it could also be possible that these systems store personal health information (PHI), which is a document containing an individual's health status and other private information.

It has been shown in several studies that making health systems interoperate will lead to long-lasting and game-changing improvements in the health sector [6]. Improvements include the rollout of a National Health Information System (NHIS) [8], a comprehensive medical history for all patients, efficiency and accuracy on the transactions performed [26][25], improved public health reporting and monitoring [24], new models of care and effective treatment [26], improved healthcare experience, and long-term economic incentives [26][25][22], among many others. To enable *interoperability* between health systems and to facilitate the exchange of healthcare data, health organizations implement a health information exchange.

A Health Information Exchange, or HIE, is a sytem for exchanging healthcare data among healthcare institutions, healthcare providers, and data repositories [8][9], which is usually implemented on a regional basis. Examples of national-level HIEs are Canada Health Infoway EHRS (CHI-EHRS) [14], Australia's National eHealth System (NEHTA) [10] (known today as the Australian Digital Health Agency [2]), and the Rwandan Health Information Exchange (RHIE) [8].

II. Related Work

A. Application of SOA in HIEs

One approach in implementing HIEs is through the adoption of a *service-oriented architecture* (SOA), where the entire HIE is composed of several physically-independent applications called *services* [11]. Each service in SOA has a unique functional context and a set of capabilities related to the context. To carry out complex business processes, the services must communicate, usually through a *middleware* software. Successful adoption of SOA leads to *intrinsic interoperability*, *federation*, *vendor diversification*, and *business and techonology domain alignment* [11]. As these goals are achieved, organizations benefit from *increased return of investment*, *increased organization agility*, and *reduced IT burden* [11].

Using SOA in implementing HIEs is evident in the CHI-EHRS and NEHTA, both of which are NHIS in developed countries. SOA is also shown to work in underdeveloped countries, as shown by the RHIE in Rwanda. With this, several countries have shown interest in applying the approach, leading to the creation of OpenHIE, an open community and technology that supports the implementation of HIEs in low-resource settings [20]. OpenHIE has already been implemented in Rwanda in 2012 (as the reference implementation) and other countries such as Tanzania, South Africa, and Bangladesh have also started the adoption of its components [5][7].

B. Performance of SOA Systems

The adoption of SOA is expected to result to a negative impact in the system's performance [19]. This is attributed to factors such as making the services communicate and

transforming the messages to comply to a common format. In order to compensate for this impact, strategies such as *horizontal scaling* and *vertical scaling* are suggested.

Several studies have already attempted to measure the performance of SOA systems and propose improvements. In [18], it is shown that several design decisions such as *asynchronous queueing* improves the performance of a SOA system, resulting to higher transactions per second (tps). Several low-level optimization techniques have also been proposed to improve the performance on middleware-based architectures, such as avoiding message reconstruction, direct transfer to network buffers, and reading the message body only when necessary [16].

C. Attribute-Based Authorization Schemes

It is noted that the nature of SOA poses a challenge to the traditional access control models [23]. Since a lot of services are involved in SOA, the access policies require richer semantics, and authorization schemes must also consider the attributes of the entities other than role and identification [27].

One way of providing an attribute-based authorization scheme is through access control policy languages such as eXtensible Access Control Markup Language (XACML) and the Security Assertion Markup Language (SAML). On this approach, access control is enforced by the system through policies defined over the attributes of the requesting entities and resources being requested.

Another way of implementing an attribute-based authorization scheme is through Ciphertext-Policy Attribute-Based Encryption (CP-ABE). In CP-ABE, access policies are embedded to the encrypted objects, and the users can only decrypt the object if the attributes embedded in their keys satisfy the embedded access policy [4]. Unlike in access control policy languages, CP-ABE does not only enforce access control upon the request of entities. It also encrypts the objects stored in the system, which keeps the objects confidential even if the storage is compromised [4]. Several HIEs have already used CP-ABE as a building block for attribute-based authorization [1], using privacy policies such as the ones found in the US Health Insurance Portability and Accountability Act (HIPAA) [17].

III. OpenHIE

OpenHIE provides a SOA-based framework for the implementation of an HIE, shown in Figure 1. The framework is composed of the following infrastructure services: *Terminology Service* (TS), *Client Registry* (CR), *Shared Health Record* (SHR), *Health Management Information System* (HMIS), *Facility Registry* (FR), and *Healthcare Worker Registry* (HWR). To facilitate interoperability, a middleware called the *Interoperability Layer* (IL) connects the services. The IL is implemented using OpenHIM [21], a SOA middleware for OpenHIE. External systems, usually called Point-of-Service (POS) systems, connect to the HIE through the IL.

The current architecture of OpenHIM is shown in Figure 2. To allow the execution of complex business processes in the

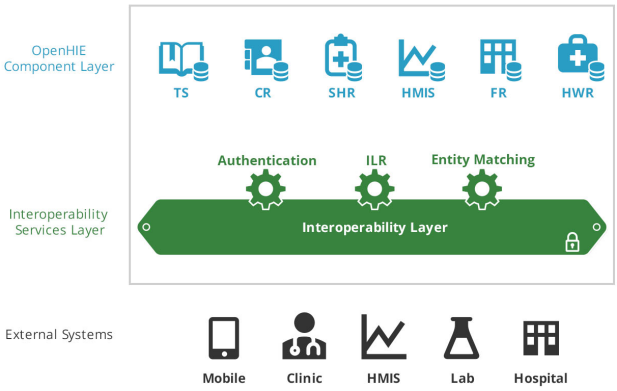


Fig. 1: High-level Architecture of the OpenHIE

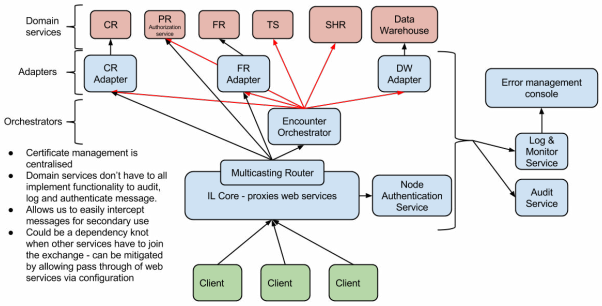


Fig. 2: The current architecture of the Interoperability Layer (OpenHIM) shown in blue [21].

HIE, adapter and orchestrator applications must be connected to the HIM. These applications, called *mediators*, provide business logic for message transformation, routing, and other complex functionalities.

OpenHIE supports internationally accepted de facto baseline protection for the privacy, security, and the confidentiality of the data that is transferred within the HIE [20]. These include mutual authentication, HTTPs connections, and transaction auditing. However, OpenHIE does not perform authorization, as it is done on the POS applications.

For consent management in sharing health information, The OpenHIE community has prescribed the "most implementable" model, due to the complexities of other models [20]. First, consent is implied to all the patients that use the HIE, but they may *opt out* (or choose to hide their PHI) if they want. Second, clinicians have *all-in* access to all of the data. Lastly, since clinicians' access to the data is all-in, *no break-the-glass* scenarios are supported. This consent management model is designed to be expanded when necessary, depending on the implementing countries and organizations.

IV. Motivation

OpenHIE v1.0 [20] provides a baseline support for data privacy. However, several areas for improvement regarding privacy can still be identified, namely *fine-grained access*

control, *security levels in data*, and *consent management*. First, OpenHIE delegates authorization to the external Point-of-Service systems, but we cannot assume that all POS applications will always authorize the users. The HIE must also be capable of authorizing *segmented access*, this is why access control must be fine-grained. Next, OpenHIE must have a mechanism for identifying the security levels of the data that is flowing through the HIE. For example, a Health Record in the SHR must have a higher security level than an hospital address in the FR. Lastly, OpenHIE's consent management model must be dynamic enough if it has to be implemented on different countries with different privacy laws.

Several means for supporting authorization are suggested in the documentation of OpenHIE. One way is to provide additional support for XACML/SAML in the IL [20]. Another way is described in the IL's design document, where authority attributes will be added in the Healthcare Worker Registries to determine transaction permissions [20]. While these approaches provide a layer of access control to the HIE, it would be desirable if the data is also encrypted [4][19], as the HIE also stores personal health information.

This study will focus on the application of Cyphertext-Policy Attribute-Based Encryption (CP-ABE) as a means of providing an authorization scheme to OpenHIE v1.0.

V. Methodology

Using CP-ABE as a building block for our authorization scheme, we can address the following privacy needs:

- Fine-Grained Access Control - All users in the HIE will have a key, and health records in the HIE are encrypted with an embedded access policy. The users can only decrypt the object if its key has the attributes that satisfy the access policy.
- Different Security Levels in Data - Security levels can be imposed to the encrypted records by adding them as attributes in the policy. Security clearances can be imposed to the users by adding them as attributes to the key.
- Consent Directive Management - By default, medical records are only accessible to their respective owner, since they have the necessary attributes. Granting access to other users can be enabled by reissuing keys with the necessary attributes to the users.
- Break-the-Glass Scenario - Implicit authorization of access to trusted users such as first-degree relatives, guardians, and partners can be enabled also by reissuing keys.

In CP-ABE, encrypting objects and embedding access policies require a master key and a private key. Decryption requires a decryption key, which is generated using a set of attributes. To enable the generation of the said keys, and to enable user and attribute registration, we introduced another service in our HIE called the *Trusted Authority* (TA).

Three instances of OpenHIE were made to check the effects of adding an ABE scheme. These HIEs will be tested by sending workloads consisting of the key workflows: *save health record*, *query health record*, and *generate decryption key*. These workflows are selected because they are common on most of the transactions in the HIE. On the save workflow, a record sent from the POS will be passed by the interoperability layer to the *clinical encounter mediator* (or *the mediator*) for validating data against the registries' data. This record will then be stored to the SHR upon successful validation. On the query workflow, the record will be queried from the SHR and it will be enriched with data from other registries before sending it to the POS. On key generation, the TA will generate a new key given the attributes sent by the user through the POS. Generating a decryption key will be an important workflow when CP-ABE comes in. A Point-of-Service application is connected to each HIE for executing the workflows.

The HIEs are simplified to only include parts that are important to the given workflows, but they still use the middleware-based architecture and it will also satisfy the save and query workflows of OpenHIE v1.0. The components of the HIEs are the POS, IL, CR, FR, HWR, SHR, the mediator, and the TA.

The first HIE is the *Default* HIE, shown in Figure 3a. On the default HIE, the save and query workflows are implemented as is. No CP-ABE schemes are implemented.

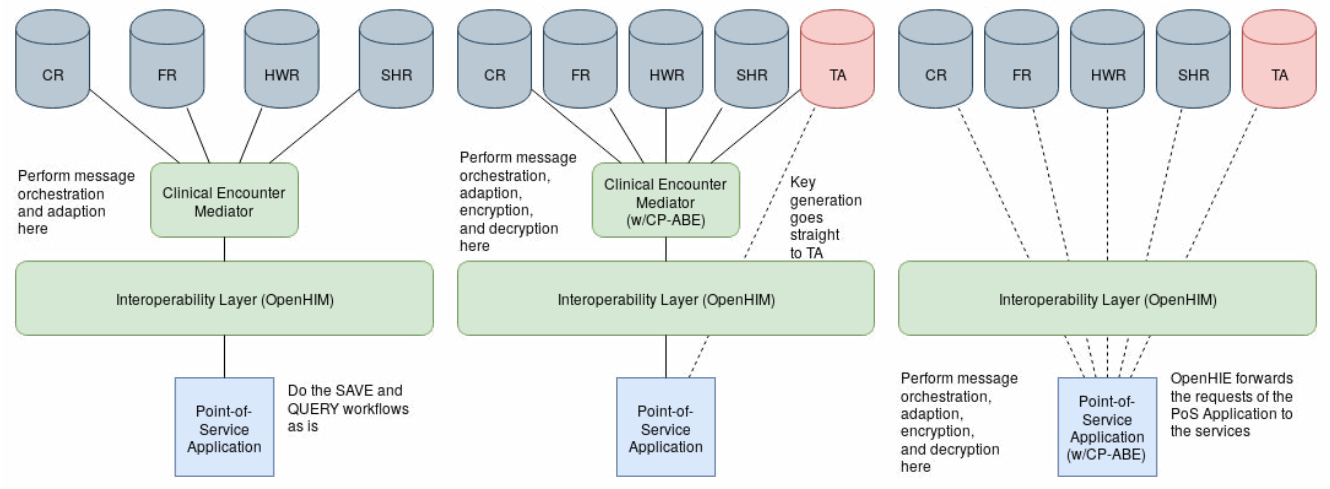
We introduce CP-ABE, along with the TA, on the second and third HIEs. On the second HIE, encryption and decryption is performed IN the mediator upon the save or query workflow. To do this, the POS will pass the decryption key to the mediator. We call the second HIE *ABE-IN*, shown in Figure 3b.

On the third HIE, encryption and decryption is performed on the POS application outside the HIE, upon the save or query workflow. For this to become possible, the complex logic in the mediator is moved to the POS application, eliminating the mediator. We call the third HIE *ABE-OUT*, shown in Figure 3c.

The implementation of ABE-IN and ABE-OUT is motivated by (1) the IL must hide all the complexities of making transactions to the HIE (ABE-IN), and (2) encryption and decryption must be performed on the side of the user, and not in the HIE (ABE-OUT).

The CP-ABE cryptosystems are implemented using the Charm framework [3], a Python framework for prototyping cryptosystems and crytographic applications. All HIE components, are implemented using Python, with the exception of the IL, where we use OpenHIM. For the database we used MySQL for the services and MongoDB for OpenHIM. All of the HIE components are designed to receive messages using a *single thread*. However, for sending messages, the mediators in *Default* and in *ABE-IN* and the POS application in *ABE-OUT* use a thread pool [12] that sends 4-5 messages simultaneously for executing business logic. All HIE components run on a local network that connects the components with a transfer rate of 100Mbps, and each HIE component runs on a Docker [13] machine that is deployed in a computer with a 3.6GHz Intel Core i7, 4GB DDR3 RAM, and 1TB HDD.

We then evaluated HIEs on the key workflows, following the processes prescribed in [15]. For each key workflow, 200



(a) Default (b) ABE-IN (c) ABE-OUT

Fig. 3: The HIEs for evaluating CP-ABE

transactions are executed on each HIE, and experiments are performed on different number of users (POS applications) and different number of attributes.

VI. Results and Discussion

We measured the transactions per second (tps) of the three HIEs under the save operation. Figure 4 compares the performance of the HIEs on the save operation with 10 attributes. On all HIEs, the number of saves per second decrease as the number of users increase. This is due to the single-threaded nature of the setup, where the IL is shared among the users and the mediator. Similar patterns can be seen on 1 attribute and 5 attributes, but with a higher tps, as the number of operations when encrypting using CP-ABE is proportional to the number of attributes.

At one user, ABE-OUT has lower tps than Default and ABE-IN because of the overhead due to the number of steps performed to serve a transaction. In the Default and the ABE-IN setup, each request to validate encounter data (there are 4) only takes one hop - from the mediator to the registry. However, in ABE-OUT, each validation request takes two hops - from the POS to IL, and from the IL to the registries. As the number of users increase, the mediator in Default and ABE-IN setup becomes a bottleneck. The transactions queue up as they wait for the mediator to encrypt and send other transactions to the SHR. In ABE-OUT, the transactions are already encrypted before they are sent to the IL, reducing the waiting time for sending other transactions. Also, there is no mediator present so there is no additional hop involved. Moreover, the requests for validation are way smaller than the encounters, which makes them faster to complete even if they introduce additional steps.

We also compared the tps of query transactions in all the HIEs. The results are similar to that of the save operation. At one user, the default HIE yields more transactions than

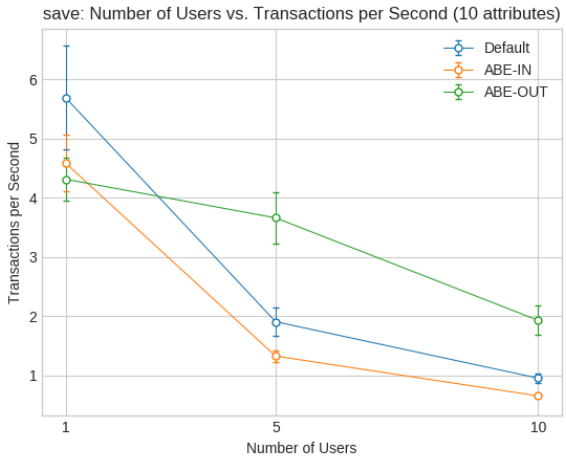


Fig. 4: Save Encounter (10 attributes): Number of Users vs Transactions per Second

ABE-IN and ABE-OUT. However, as the number of users increase, ABE-OUT performs better than default and ABE-IN. The reason for this behavior is the same.

Figure 5 shows that the tps in the key generation of ABE-IN decreases as the number of users and the number of attributes increase. This is expected since the TA runs on a single thread and the CP-ABE performs more operations as the number of attributes increase. The number of key generations per second in ABE-OUT has a similar result since the workflow for key generation is the same for both setups.

The space consumed in the SHR is also measured per save transaction. HIEs with CP-ABE consume more space in the SHR as encounters are saved. This is expected, as CP-ABE encryption appends a value for each attribute in the access policy [4], making the ciphertext larger as the number of

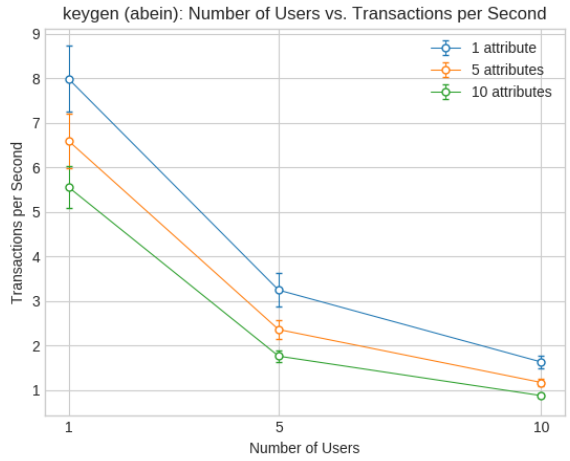


Fig. 5: ABE-IN Key Generation: Number of Users vs. Transactions per Second

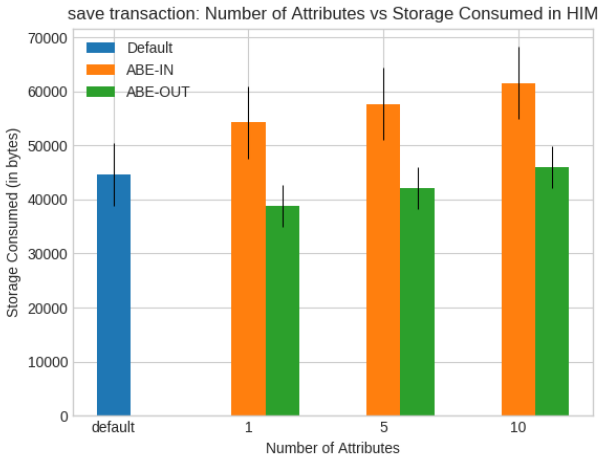


Fig. 6: Save Encounter: Space Consumed in IL vs Number of Attributes

attributes increase. The conversion of ciphertext to base64 format for storage also contributes to larger record sizes. The space consumed in the TA on key generation is also measured, and it is straightforward that more attributes used will lead to larger space consumed.

Figure 6 shows the space consumed in IL for auditing on the save encounter operation for all HIEs. It is clear that the increasing the attributes in ABE-IN and ABE-OUT also increases the space consumed in the IL. It is expected, as the ciphertext gets larger as the number of attributes in the access policy increase. It can also be noted that save and query transactions consume less space in ABE-OUT as compared to ABE-IN. The large space overhead in ABE-IN and in the Default is due to the programming convention that mediators must always return metadata to the IL [21]. The mediators can be programmed to return less, but their implementation is not modified for now.

It is also expected that key generation transactions consume more space in the IL when the number of attributes increase. By following the key generation algorithm of CP-ABE [4], we can say that more attributes lead to more space consumed in the IL.

Lastly, a comparison on the throughput from POS to IL in the save operation shows that the ABE-OUT POS sends more bytes, as compared to Default POS and ABE-IN POS which only sends the unencrypted record, along with the decryption key and access policies for ABE-IN. Figure 7 show the comparison when 10 attributes is used in saving records. We can say that the ABE-OUT POS utilizes more bandwidth since transactions finish earlier and more HTTP requests and more data are sent per transaction. For the query transaction, a similar pattern can be observed.

VII. Conclusions and Recommendations

We evaluated the application of using CP-ABE as a building block for implementing an authorization mechanism within

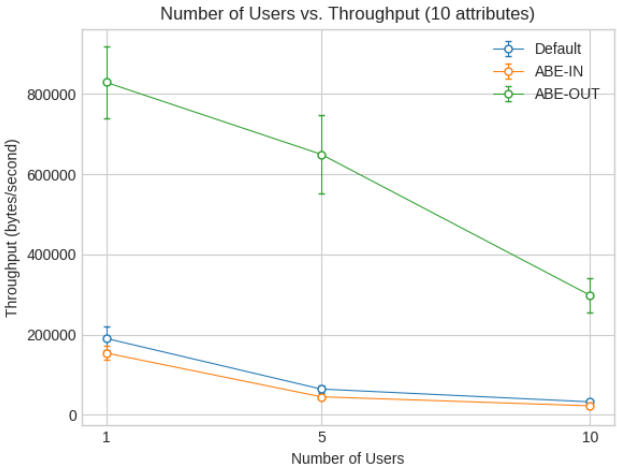


Fig. 7: Bandwith use on the Save transaction for 10 Attributes

OpenHIE. Different instances of OpenHIE were deployed to analyze its the effect on different parts of the HIE. Changes in the workflows of the HIE are required when a CP-ABE based authorization scheme will be implemented.

Once implemented, several quantifiable changes can be observed. First, the tps decrease as the number of attributes used for encryption increase. Also, on all HIEs, the tps and the bandwidth utilized decrease as the number of users increase. This is because the IL gets shared among the registries and the PoS systems. It is recommended that on a production environment, the HIE components must be scaled vertically or horizontally, using more than one thread not only to utilize the resources available but also to serve requests faster.

The application of CP-ABE also affects the space consumed by the transactions in the IL, in the SHR, and in the TA. More attributes used would lead to larger ciphertext messages, larger encryption polices, and larger decryption keys. Hence,

the storage consumed will be larger. It is recommended that the storage used by the TA, SHR, and IL, be scaled out if ever CP-ABE will be used. As seen from the results, the HIEs with mediators consume a lot of space in the IL for every transaction, as compared to ABE-OUT. While it is a convention in OpenHIE that the mediators must provide audit data to the IL, a more efficient way of getting audit info while spending less space must be studied.

We have also seen that, among the three HIEs, ABE-OUT performs better than the default setup when the number of users increase. For only one user, the default HIE performs better. The increasing overhead in the default HIE as the number users increase is due to the bottleneck which is the mediator. It must be noted, however that this holds for CP-ABE schemes that only use up to 10 attributes.

The results show that for 1-10 attributes in the encryption keys and the policies, HIEs with CP-ABE provide an overhead over the default setup. On all HIEs, the performance issues can be answered by scaling out the resources of the components. The implementer is free to choose which setup has to be implemented. The implementer can choose ABE-IN, which provides a lower barrier of entry to connecting systems, but has a larger overhead when it comes to the save and query operations. The implementer can also choose to adopt ABE-OUT, which has faster transaction times, higher bandwidth utilization, higher control over encryption/decryption, but has a higher barrier of entry as business logic must also be implemented in the connecting systems. It is inevitable that there will be an overhead in the performance as the HIEs follow a service-oriented archtiecture [19]. With this in mind, it is recommended that the techniques introduced in [18] and [16] be considered in implementing a SOA-based HIE system.

REFERENCES

[1] Samydurai A, Revathi K, Prema P, Arulmozhiarasi D S, Jency J, and Hemapriya S. Secured health care information exchange on cloud using attribute based encryption. In *Signal Processing, Communication and Networking (ICSCN), 2015 3rd International Conference on*, pages 1–5, March 2015.

[2] Australian Digital Health Agency. About the agency - australian digital health agency. Accessed: 2016-09-28.

[3] Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, Jun 2013.

[4] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 321–334, May 2007.

[5] Paul Biondich, Shaun Grannis, and Christopher J. Seebregts. Open health information exchange (OpenHIE): An international Open-Source initiative in support of large scale data interoperability for the under-served. In *American Medical Informatics Association Annual Symposium*, 2013.

[6] David J. Brailer. Interoperability, the key to the future health care system. *Health Affairs*, January 2005.

[7] Ryan Crichton. *The Open Health Information Mediator: an Architecture for Enabling Interoperability in Low to Middle Income Countries*. Dissertation, University of KwaZulu-Natal, Durban, 2015.

[8] Ryan Crichton, Deshendran Moodley, Anban W. Pillay, Richard Gakuba, and Christopher J. Seebregts. An architecture and reference implementation of an open health information mediator: Enabling interoperability in the rwandan health information exchange. In *Foundations of Health Information Engineering and Systems - Second International Symposium, FHIES 2012, Paris, France, August 27-28, 2012. Revised Selected Papers*, pages 87–104, 2012.

[9] Brian E. Dixon, Atif Zafar, and J. Marc Overhage. A framework for evaluating the costs, effort, and value of nationwide health information exchange. *J Am Med Inform Assoc*, 17(3):295–301, Jul 2010. amiajnl570[PII].

[10] National eHealth Transition Authority (NEHTA). National e-health transition authority. nehta blueprint - blueprint v2.0. https://www.digitalhealth.gov.au/implementation-resources/ehealth-foundations/nehta-blueprint. Accessed: 2016-09-07.

[11] Thomas Erl. *SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.

[12] Python Software Foundation. 17.4. concurrent.futures launching parallel tasks python 3.5.5 documentation. https://docs.python.org/3.5/library/concurrent.futures.html. Accessed: 2018-05-23.

[13] https://www.docker.com. Docker - build, ship, and run any app, anywhere. https://www.docker.com. Accessed: 2018-05-22.

[14] Canada Health Infoway. Canada health infoway. electronic health record solution (ehrs) blueprint v2 summary. https://www.infoway-inforoute.ca/en/component/edocman/resources/technical-documents/390-ehrs-blueprint-v2-summary?Itemid=101. Accessed: 2016-09-07.

[15] Raj Jain. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. Wiley professional computing. Wiley, 1991.

[16] Hiranya Jayathilaka, Pradeep Fernando, Paul Fremantle, Kasun Indrasiri, Dushan Abeyruwan, Supun Kamburugamuve, Sadeep Jayasumana, Sanjiva Weerawarana, and Srinath Perera. Improved server architecture for highly efficient message mediation. In *Proceedings of International Conference on Information Integration and Web-based Applications & Services*, IIWAS '13, pages 418:418–418:427, New York, NY, USA, 2013. ACM.

[17] Peifung E. Lam, John C. Mitchell, Andre Scedrov, Sharada Sundaram, and Frank Wang. Declarative privacy policy: Finite models and attribute-based encryption. In *Proceedings of the 2Nd ACM SIGHIT International Health Informatics Symposium*, IHI '12, pages 323–332, New York, NY, USA, 2012. ACM.

[18] Nariman Mani, Dorina C. Petriu, and Murray Woodside. Towards studying the performance effects of design patterns for service oriented architecture. In *Proceedings of the 2Nd ACM/SPEC International Conference on Performance Engineering*, ICPE '11, pages 499–504, New York, NY, USA, 2011. ACM.

[19] Liam O'Brien, Paulo Merson, and Len Bass. Quality attributes for service-oriented architectures. In *Proceedings of the International Workshop on Systems Development in SOA Environments*, SDSOA '07, pages 3–, Washington, DC, USA, 2007. IEEE Computer Society.

[20] OpenHIE.org. Openhie. http://ohie.org/. Accessed: 2016-05-28.

[21] OpenHIM.org. Openhim - simplifying interoperability. http://openhim.org/. Accessed: 2016-05-28.

[22] Hayoung Park, Sang il Lee, Hee Hwang, Yoon Kim, Eun-Young Heo, Jeong-Whun Kim, and Kyooseob Ha. Can a health information exchange save healthcare costs? evidence from a pilot program in south korea. *International Journal of Medical Informatics*, 84(9):658 – 666, 2015.

[23] Pierangela Samarati and Sabrina Capitani de Vimercati. *Access Control: Policies, Models, and Mechanisms*, pages 137–196. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[24] Jason S. Shapiro. Evaluating public health uses of health information exchange. *Journal of Biomedical Informatics*, 40(6, Supplement):S46 – S49, 2007. Developing Common Methods for Evaluating Health Information Exchange.

[25] Peter Sprivulis, Jan Walker, Douglas Johnston, Eric Pan, Julia Adler-Milstein, Blackford Middleton, and David W Bates. The economic benefits of health information exchange interoperability for australia. *Australian Health Review*, 31(4), November 2007.

[26] Jan Walker, Eric Pan, Douglas Johnson, Julia Adler-Milstein, David W. Bates, and Blackford Middleton. The value of health care information exchange and interoperability. *Health Affairs*, January 2005.

[27] E. Yuan and J. Tong. Attributed based access control (abac) for web services. In *IEEE International Conference on Web Services (ICWS'05)*, page 569, July 2005.