Feasibility of Homomorphic Encryption for Sharing I2B2 Aggregate-Level Data in the Cloud

Jean Louis Raisaro, MS[1], Jeffrey G Klann, PhD[2,3,4], Kavishwar B Wagholikar, MD PhD[2,3,4], Hossein Estiri, PhD[2,3,4], Jean-Pierre Hubaux, Dr-Eng[1] and Shawn N Murphy, MD, PhD[2,3,4]

[1]École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland; [2]Partners Healthcare, Boston, MA, USA; [3]Harvard Medical School, Boston, MA, USA; [4]Massachusetts General Hospital, Boston, MA, USA

Abstract

The biomedical community is lagging in the adoption of cloud computing for the management of medical data. The primary obstacles are concerns about privacy and security. In this paper, we explore the feasibility of using advanced privacy-enhancing technologies in order to enable the sharing of sensitive clinical data in a public cloud. Our goal is to facilitate sharing of clinical data in the cloud by minimizing the risk of unintended leakage of sensitive clinical information. In particular, we focus on homomorphic encryption, a specific type of encryption that offers the ability to run computation on the data while the data remains encrypted. This paper demonstrates that homomorphic encryption can be used efficiently to compute aggregating queries on the ciphertexts, along with providing end-to-end confidentiality of aggregate-level data from the i2b2 data model.

Introduction

With the increasing digitalization of medical information, the ability to share data across clinical sites is indispensable in realizing the full promise of personalized medicine. Yet, because of privacy and security concerns, sharing clinical data across healthcare organizations is currently extremely difficult or impossible to realize. The number of data breaches is on the rise and there is significant public pressure[1–3] for clinical sites to not share data stored in their repositories to ensure that patients' information is adequately protected. As a result, the privacy risks associated with medical data sharing represent a real barrier to medical advancement, if effective privacy-preserving technologies are not adopted.

Existing tools designed to enable collaborative medicine, such as Informatics for Integrating Biology and the Bedside (i2b2)[4] and TranSMART[5], lack the necessary security and privacy guarantees to be applied in untrusted environments. Their current deployments pose significant challenges in terms of both security (access control, accountability, data traceability) and privacy (trust management, confidentiality and resilience against inference attacks) that need to be addressed in order to reassure clinical sites and foster clinical data-sharing. The evolution of the regulation towards further guarantees (e.g., HIPAA in USA and the new GDPR in EU) also reflects these urgent needs.

In the last few years, the IT security and privacy community has made substantial progress in the development of new sophisticated tools, also known as "privacy-enhancing technologies (PETs)", that aim at responding to these needs. PETs provide strong security and privacy guarantees for the management of sensitive data[6]. Yet, due to the lack of awareness and their inherent complexity, the biomedical community has largely left PETs unutilized. So far, only a few isolated PETs-based systems have been deployed in medical operational settings[7–10]. Therefore, it is essential to bridge this cultural gap, in order to address the privacy and security concerns affecting the medical community.

This paper addresses this challenge by demonstrating the use of PETs in clinical environments. We explore the feasibility of using emerging technology for secure computation, specifically *homomorphic encryption*, for securely and privately sharing aggregate-level data from the most widespread data model for clinical research,

notably *i2b2*. Particularly, we introduce a new model that efficiently uses homomorphic encryption to enable clinical sites to share i2b2 aggregate-level data with multiple researchers through an untrusted cloud provider without having to worry about potential leakages of sensitive information to untrusted third parties.

i2b2 aggregate-level data consists of patient counts for unique combinations of location, ontology concept and time, e.g., the total number of patients at site S_i that got "Type I Diabetes" observed in the first quarter of 2017. Although aggregate-level data is not directly identifying, sites willing to share the data by using a public cloud, are concerned about unintended leakages of sensitive information to untrusted third parties. For example, hospital-specific utilization patterns can reveal hospital operations and budgeting, and aggregates with small counts can reveal individual patients[11]. Indeed, the leakage of this information can severely damage sites' reputation and jeopardize the privacy of their patients. Our solution ensures end-to-end protection of data confidentiality against any third party not in possession of the decryption key. Only a legitimate and authorized investigator can see the end result of a query while the confidentiality of the original data is always preserved.

Informatics for Integrating Biology and the Bedside (i2b2)

i2b2 is an open source clinical platform for enabling secondary use of electronic health records (EHR) used at over 150 clinical sites and covering more than 250 million patients' records only in the US[4]. I2b2 is designed to enable investigators to perform queries on an enterprise data-repository in order to find sets of patients that would be of interest for further clinical research studies. It consists of a simple and flexible relational data-model based on a "star schema" and a set of server-side software modules, called "cells", which are responsible for the business logic of the platform and are organized in a "hive". The data model stores, in a narrow table called "*observation fact"* table, clinical observations (or "facts") about patients such as diagnoses, medications, procedures, and demographics, along with a date, a patient identifier and an encounter identifier. Each observation is encoded by an ontology concept from a medical terminology, such as the International Classification of Disease (ICD) or the US National Drug Code (NDC). The use of extendable ontologies makes i2b2's model highly adaptable to site-specific coding and easily deployable on top of existing EHR systems. Besides the observation fact table, there are four other "dimension" tables that further describe patients' data and meta-data. Queries are built in a Web-based query tool by combining ontology codes, organized in a hierarchical tree-based structure, with logical ORs and ANDs operators. i2b2 is supported by an active community and has become one of the most popular open-source prjects in health care research.

Homomorphic Encryption

Homomorphic encryption is a special form of public-key encryption that enables computations on encrypted values (see Figure 1). As with every public-key encryption scheme, it consists of two algorithms: an encryption algorithm and a decryption algorithm. Encryption and decryption can be seen as inverse operations. At one end, the encryption algorithm takes as input a plaintext message m and a public cryptographic key K and outputs a ciphertext c. On the other end, the decryption algorithm takes as input the ciphertext c and a secret cryptographic key k and outputs the original message m.

In general, the public key K is derived from a secret key k in such a way that the inverse operation is not possible. We can say that an encryption scheme is secure if the ciphertext does not reveal any information about the underlying message to anyone without the secret key k or, similarly, if anyone without the secret key k is unable to distinguish an encryption of a message m_0 from the encryption of another message m_1.

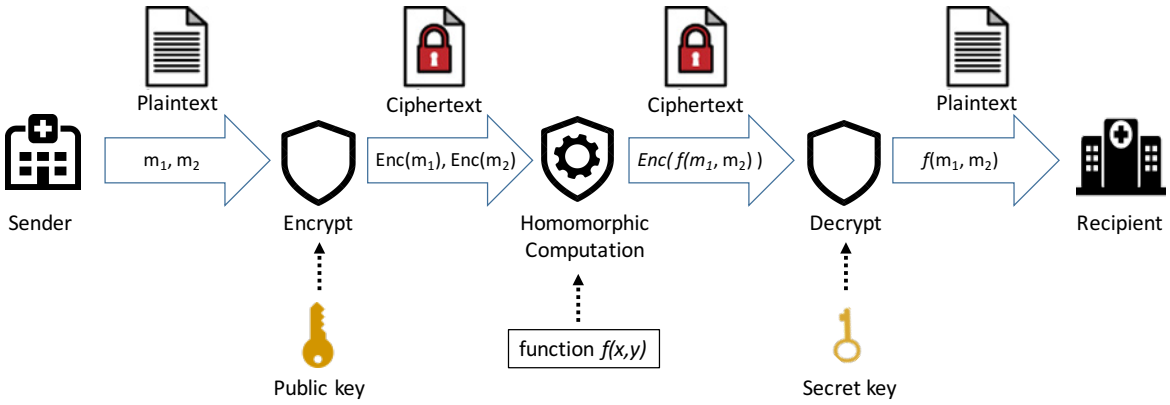

Figure 1. High-level representation of homomorphic encryption enabling computations on encrypted data. Different keys are used to encrypt and decrypt messages.

In contrast to standard public-key encryption that requires decryption in order to carry out any computation on the original messages, homomorphic encryption allows to perform computations directly on the encrypted messages. In fact, homomorphically-encrypted messages can be combined together, thus generating an encrypted result that, when decrypted, yields the result of a meaningful operation (like addition or multiplication) on the original messages. For example, by using the homomorphic encryption scheme ElGamal on elliptic curves[12], also denoted as EC ElGamal, one can decrypt the sum of two ciphertexts and obtain the result of the sum of the two original messages.

Thanks to this property, homomorphic encryption schemes enable several use cases such as performing analytics in an untrusted environment (e.g., a public cloud) while ensuring the confidentiality of the data processed. Yet, it is important to note that homomorphic encryption introduces sometimes unpractical costs. For example, f*ully* homomorphic encryption allows any computation to be carried out on the ciphertext but at the cost of unacceptable storage and computational overheads. Instead, p*artially* homomorphic encryption has less flexibility than fully homomorphic encryption, as it allows only some specific computations to be carried out the ciphertext, but with reasonable storage and computation overheads. In this paper, we explore how to use *additively* (hence partially) homomorphic encryption, i.e., encryption that supports only additions on ciphertexts, to enable the sharing of aggregate-level data from the i2b2 data model.

Methods

Before introducing the proposed solution, we model the system and the threats that our solution aims at preventing.

System and Threat Models

We consider the data-sharing scenario depicted in Figure 2 where multiple clinical sites use i2b2 for internal purposes and want to share i2b2 aggregate-level data with multiple investigators by the means of a central public cloud. Several *Clinical Data Research Networks (CDRNs)* in the US *National Patient-Centered Clinical Research Network (PCORNet)*[13] are concrete examples of such a data-sharing scenario. Actually, there is an increasing need to shift from decentralized data-sharing models to more centralized ones, because of insufficient resources (both human and technical) at sites to maintain an interoperable network.

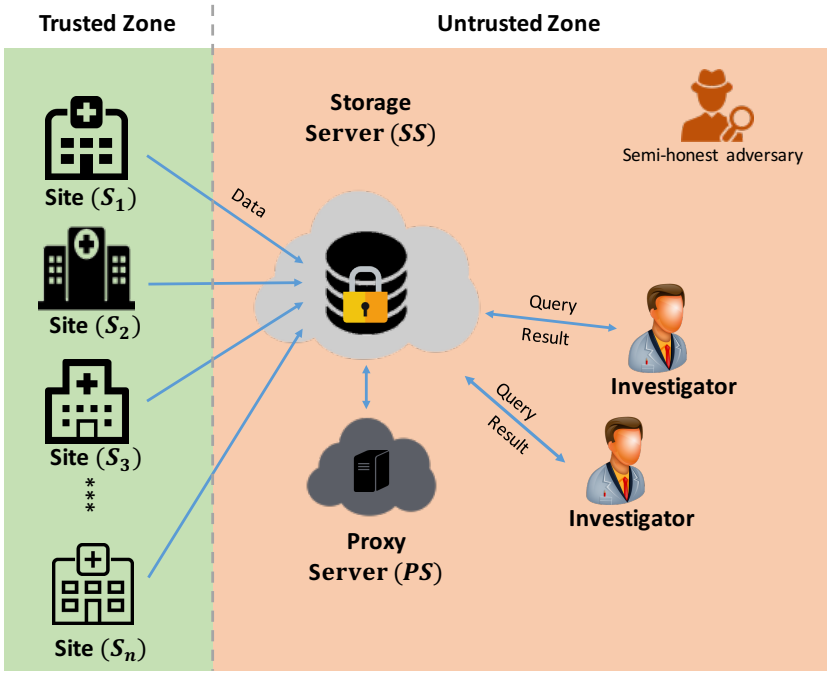

Figure 2. High-level representation of a centralized data-sharing scenario with: (i) a system model including several clinical sites, two distinct cloud providers and one or multiple investigators and (ii) a threat model including honest-but-curious adversaries.

In this setting, the standard system model usually consists of three main parties: (i) one or multiple clinical *sites* (S_i), organized in a network or federation, that aim at sharing their data through the cloud, (ii) one or multiple *investigators* willing to access this information, and (iii) a *storage* server (*SS*) hosted on the cloud and responsible for the central storage of sites' data and for responding to investigators' queries. Yet, in order to achieve strong security and privacy guarantees, we need to introduce an additional party on top of this model, namely a new server called *proxy* server (*PS*). The *proxy* server is necessary for helping the *storage* server during the privacy-preserving data-sharing protocol and ensuring trust decentralization. Trust decentralization is a crucial requirement for this kind of data-sharing scenarios as it allows to avoid single points of failure in the system. The storage and proxy servers can be represented by any pair of independent servers such as, for example, two servers at two distinct public cloud providers such as AWS Cloud[14] or Google Cloud[15]. It is crucial that these servers are independent and do not collude, otherwise the system could be easily compromised.

In this system model, we consider a *semi-honest* adversarial setting or threat model. This means that all the parties except the clinical sites, that are the legitimate owners of the data, are supposed to honestly follow the data-sharing protocol but might try to passively infer extra information from what they see during the protocol. For example, an insider or a hacker that has bypassed the cloud provider's firewalls and tries to passively infer sensitive information from the data stored at *SS,* can be considered as a semi-honest adversary. Similarly, an investigator that legitimately queries the system with a series of carefully crafted requests in order to infer sensitive information about patients at the different sites, can also be considered as a semi-honest adversary. In contrast, an adversary that would try to cheat and depart from the protocol would be considered as *malicious*. Although, in most of the cases, adversaries are considered to be malicious, we note that the assumption of a semi-honest adversarial setting is reasonable in practice, especially in the medical context, as the different parties involved in the data-sharing process have either business/operational or reputational incentives to honestly follow the protocol and to not cheat. Hence, for the rest of the paper, we ignore the case of a malicious active adversary.

Data Model

In order to store i2b2 aggregate-level data, we designed a data model that is fully compatible with the i2b2 data model and consists of a single table with three aggregating attributes and one counting attribute (see Table 1). Each row of this table stores the total number of patients for whom the same observation (unique combination of location, ontology concept and time) has been registered in the i2b2 *observation fact* table of any of the sites. In particular, the location column stores the information about the site at which the observation has been collected. The ontology concept column stores the unique identifier (e.g., concept code or concept path from the hierarchical ontology tree) of the condition, medication, lab result, or procedure that has been observed. The time column stores the time unit at which the observation has been collected, e.g., a particular month, quarter, year. Finally, the count column stores the total number of patients with the observation specified by the ontology concept *X* and collected at location *Y* and time *Z*.

Table 1. Proposed data model storing i2b2 aggregate-level data.

Patients Totalnum	
PK	**Location**
PK	**Ontology_Concept**
PK	**Time**
	Count

This data model provides a high degree of flexibility as the three aggregating attributes can be defined at any spatial level, time scale and ontology concept of interest. Typical queries to this data model are of the form

```
SELECT */SUM(count)
FROM patients_totalnum
WHERE (ontology_concept=X1 OR ontology_concept=X2 OR …) AND
(location=Y1 OR location=Y2 OR …) AND (time=Z1 OR time=Z2 OR …)
GROUP BY *;
```

where X_i, Y_i and Z_i denote any possible value in the corresponding column. A concrete example of such queries could be: "What is the total number of patients per site that had been diagnosed with HIV in 2015?". Our privacy-preserving model enables queries of this type on homomorphically encrypted data.

Privacy-Preserving Data-Sharing Protocol

The proposed privacy-preserving data sharing protocol is described in Figure 3 and it is based on the additively-homomorphic properties of the EC ElGamal cryptosystem[16].

Initialization phase. We begin with the initialization phase. During this phase, the *storage* and the *proxy* servers, both equipped with a processing unit named "*crypto engine*", independently generate a pair of asymmetric cryptographic keys (public and secret keys) for the EC ElGamal cryptosystem and combine their public keys to obtain a public *shared* key (black key in Figure 3). The two servers store their secret keys safely and do not exchange them with any other party in the system.

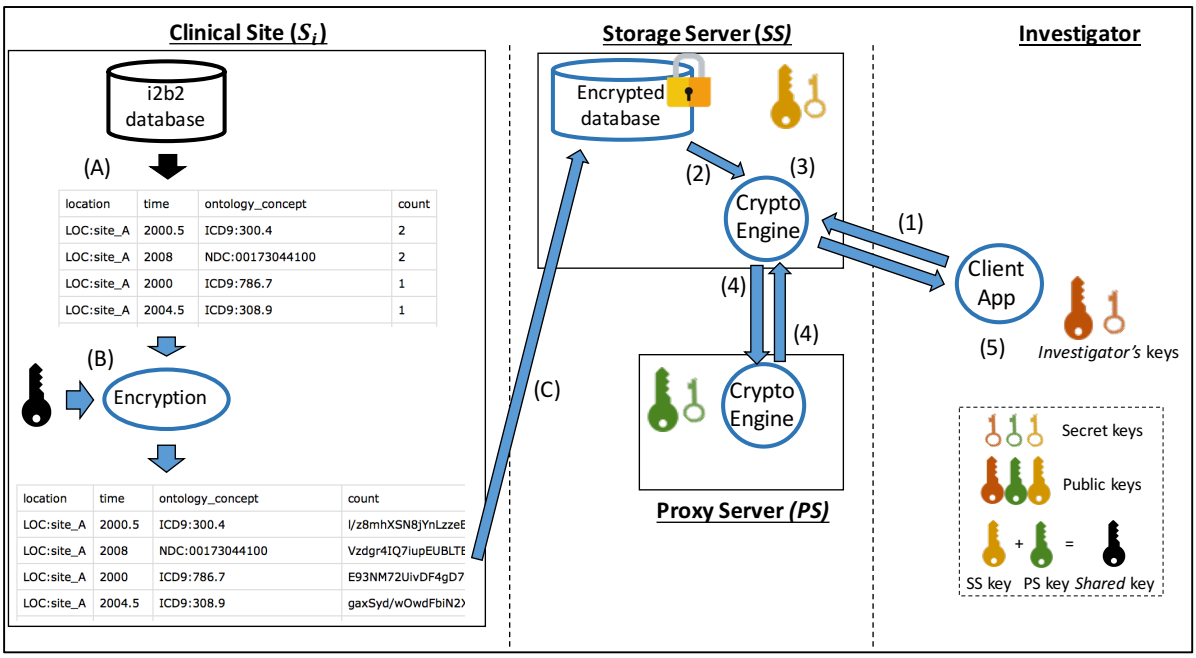

Figure 3. High-level representation of the proposed privacy-preserving data-sharing protocol. Steps (A-C) represent the ETL phase whereas steps (1-5) represent the secure query processing.

Extract, transformation and loading (ETL) phase. In the ETL phase, the *shared* public key is used by each site to encrypt their i2b2 aggregate-level. More specifically, each site extracts plaintext aggregate-level data from their i2b2 local databases, transform it in order to fit the aggregate data model, encrypts the total numbers of patients and uploads the encrypted table to the *storage* server in the cloud. Encryption under the shared key ensures that as long as an adversary does not obtain the secret keys of both the *storage* and *proxy* servers, the confidentiality of the data is preserved. In this way, we avoid a single point of failure in the system as both cloud providers must be compromised simultaneously in order, for an adversary, to reconstruct the *shared* secret required to decrypt the encrypted data stored at the *storage* server.

Querying phase. Once the encrypted data is securely stored on the *storage* server, an investigator can query it by specifying the combination of aggregating attributes values for which he wants to receive the patients' total numbers. As such, the investigator generates a temporary pair of asymmetric cryptographic keys (public and secret keys in red in Figure 3) and sends the query along with his public key to the *storage* server (step 1 in Figure 3). The temporary pair of keys is freshly generated for each new query. Once the query received, the *storage* server fetches from the database the records, encrypted under the *shared* public key, that satisfy the query criteria (step 2 in Figure 3) and uses the *crypto engine* to homomorphically aggregate them (step 3 in Figure 3).

Due to the additively-homomorphic properties of the EC ElGamal encryption scheme, records can be aggregated without the need of decryption, thus yielding an encrypted result. We note that a *shared* secret key corresponding to the *shared* public key does not exist in the system and decryption would not be possible unless the *storage* and *proxy* servers collude, which is highly unlikely based on the above-mentioned system model. Yet, in order to be decrypted by the investigator, the result encrypted under the *shared* public key must be re-encrypted under the investigator's public key. Therefore, the *storage* server starts an interactive two-party *re-encryption* protocol with the *proxy* server in order to switch the encryption of the result (step 4 in Figure 3) so that it is decryptable only by the investigator issuing the query. As such, each server sequentially uses its secret key and the investigator's public key to partially modify the encryption of the result.

At the end of the protocol, the *storage* server obtains the query result encrypted under the investigator's key. Note that during the *re*-encryption protocol, none of servers can see the query result in the clear. Optionally and

depending on the trust level of the investigator, the *storage* server can then homomorphically add random Laplacian noise to obfuscate the query result and achieve *differential privacy*[17,18] in order to mitigate the risk of re-identification of a patient in one of the sites' databases.

Finally, the (potentially obfuscated) result encrypted under the investigator's public key is sent back to the investigator who can decrypt it by using his secret key (step 5 in Figure 3) and visualize the result. We note that the use of differential privacy implies that during the system initialization phase, each investigator is assigned a differential privacy budget "ε" that is progressively reduced each time a new query is performed. When the budget is zero, the investigator cannot perform queries anymore.

Results

Prototype Implementation

We implemented the proposed solution as a client-server application in the Go language (version 1.8) by relying on the open-source ONet framework for decentralized and distributed protocols[19]. For the cryptographic operations in the privacy-preserving data-sharing protocol, we relied on the Go's native cryptographic libraries and the ONet advanced cryptographic library[19] that includes an implementation of the EC ElGamal on the Ed25529 elliptic curve[12] with 128-bit security.

The source code of our application is publicly available at https://github.com/JLRgithub/PDCi2b2. The client component enables the user to encrypt aggregate-level data extracted from the i2b2 data model and to specify the desired combination of aggregating attributes in order to query the encrypted data stored in a relational database system on the *storage* server. The server component enables to setup the *storage* and *proxy* servers, generate the cryptographic keys and execute the functionalities of the "*crypto engine*", i.e., the homomorphic aggregation, the two party *re-encryption* protocol and the homomorphic obfuscation.

Performance Evaluation

We tested the performance of the proposed model in a real cloud-based environment. We set up the *storage* and *proxy* servers on two separated AWS instances both running 64-bit Linux Amazon-edition on a dual-core Intel Xenon E5645 processor with 2.40GHz frequency and 4GB RAM. Note that for an ideal deployment the two instances must be hosted by two different cloud providers in order to minimize the risk of collusion. We used an off-the-shelf laptop running Mac OS X on Intel Core i7 processor with 3.1GHz frequency and 16GB ram as the client machine.

We simulated three different sites and, for each of them, we encrypted patients' total numbers extracted from the i2b2 demo dataset. We considered only concepts in the lowest level of the ontology tree-based hierarchy, i.e., the leaves, and we used quarters as time unit. Patient counts for ontology concepts at higher levels in the tree-based hierarchy can be obtained by recursively aggregating patient counts for the children concepts. Therefore, we used the concept path to uniquely identify each ontology concept in order to keep track of all the parental levels in the ontology tree. The resulting database had 10,767 rows per site for a total of 32,301 rows. We store the encrypted data in a PostgreSQL database on the *storage* cloud server.

We measured the time of each cryptographic operation used in the privacy-preserving data-sharing protocol by averaging over 100 random queries for different levels of aggregations and different combinations of ontology concepts, times and locations. We used a single thread of execution in all of our experiments, not taking advantage of the available parallelism, to obtain a lower bound on the performance of the proposed system. Table 2 shows the amortized times obtained in our experiments. We note, however, that these operations are naturally parallelizable hence performance can be significantly improved.

Table 2. Amortized times in milliseconds for the cryptographic operations run in the proposed privacy-preserving data-sharing protocol.

Cryptographic Operation	Amortized Time per Record (ms)
Homomorphic aggregation	0.004
Proxy Re-Encryption	13.77
Decryption	0.9

Homomorphic aggregation scales linearly with the number of records that satisfy the querying criteria, whereas the *re-encryption* and the *decryption* operations scale linearly with the number of query results. As an illustrative example, an aggregate query asking the total number of patients for whom any diabetes-related concepts have been observed in the first quarter of 2012 across all sites takes, on average, a total of 807 ms, where 214 ms is the time necessary to fetch the encrypted records satisfying the query (121 ontology concepts per site contain the keyword "diabetes" in their ontology path) from the database, 1.4 ms are used to homomorphically aggregate the records, 13 ms are used in the *re-encryption* protocol and 0.9 ms to decrypt the result. The remaining time, 576 ms, is the communication time between the client and the *storage* server necessary for sending the query and receiving the encrypted result, which was constant for all tested queries. Contrarily, a non-aggregate query that does not involve homomorphic aggregation, such as the total number of patients at site 1 for whom Diabetes Mellitus (ICD9:250) has been observed in the first quarter of 2012, takes only 590 ms in total (0.58 ms to fetch the record from the database, 13 ms for re-encryption, 0.9 ms for decryption and 576 ms for communication).

Discussion

In this paper, we demonstrated the feasibility of using homomorphic encryption in order to enable the sharing of i2b2 aggregate-level data through the cloud. We showed how this type of privacy-enhancing technology can be efficiently used to enable new use cases that are currently impossible due to technical or legal reasons, or need to rely on costly and time-consuming ethical and legal processes. Our solution introduces a new and simple model that minimizes the risk of unintended leakages of sensitive information by ensuring end-to-end confidentiality protection with respect to untrusted third parties. Results obtained in a basic testing environment are promising and show reasonable overhead especially from a computational point of view. The storage overhead is also practical as an encrypted database is only four times larger than its unencrypted version.

Yet, it is important to mention that a main limitation of the proposed model is availability— both the cloud servers (*storage* and *proxy* servers) need to be online in order for the system to be operational. Although this might slow down adoption in an operational setting, replication can be used to mitigate this problem. Actually, it would be sufficient to replicate the number of instances of the same servers at the two cloud providers in order to guarantee service continuity: if one instance is unavailable, another one can immediately replace it.

We emphasize on how the presence of two servers is necessary for the overall security of the system in order to achieve trust distribution and avoid a single point of failure in the system. If the cloud provider hosting the *storage* server was compromised, the data would still be secure because the attacker would need to also compromise the cloud provider hosting the *proxy* server in order to reconstruct the *shared* decryption key necessary to decrypt the data. Differently, if the security of the whole system was centralized on a single party (e.g., only on the *storage* cloud), it would be easier for the attacker to concentrate all his resources on a single target thus increasing the likelihood of jeopardizing the whole system. Intuitively, the larger the number of parties among which the trust is distributed (i.e., the number of *proxy* servers) the more secure is the system. Systems such as UnLynx[20] or Sharemind[21] rely on this idea. However, we believe that two independent servers can represent a practical and acceptable compromise between security and system usability.

Moreover, we note that the encryption scheme used (EC ElGamal) is only additively homomorphic, hence it enables only linear aggregations in the encrypted domain (e.g., summations, counts). For more complex types of

aggregation (e.g., correlations, linear regressions) a *somewhat* homomorphic encryption scheme, such as the FV encryption scheme[22], that also enables multiplications on encrypted data can be used instead of EC ElGamal at the expense of an increased storage overhead, without modifying the proposed model. Although we used homomorphic encryption, there are other privacy-enhancing technologies that could also be used in this context. Secure computation is well supported also by *garbled circuits*[23], as well as by *linear secret sharing*[24]. Yet, these two technologies require an increased level of interaction between the different parties than homomorphic encryption, thus resulting in even less reliable models from an operational perspective.

In conclusion, the proposed solution based on homomorphic encryption has been shown to be efficient and secure in the *semi-honest* adversarial setting as sites' sensitive data remains encrypted during the whole data-sharing process. Only the investigator sending the query is able to decrypt the end result. In this paper, we did not address the case of a *malicious* active adversary who tampers with the data and the protocol in order to infer sensitive information as it would be in contrast with the business incentives of a cloud provider. Yet, we note that our model can be easily extended to this case in order to protect also the integrity of the data and the computations carried out in the cloud. Techniques such as *zero-knowledge proofs of correctness*[25], can be added on top of our encryption scheme in order to ensure the integrity of the different steps of the data-sharing protocol and the identification of misbehaving parties at the expenses of an increased computational cost. These proofs can be securely stored on a distributed ledger, e.g., a blockchain, and be verified by any party in the system. We will consider this enhancement in future work.

Conclusion

Privacy and security concerns represent real obstacles for medical progress. Bridging the gap that still exists between the medical community and the IT security and privacy community is key for addressing this problem and making new and sophisticated privacy-enhancing technologies be core components of clinical operational tools in the near future. Although homomorphic encryption is still at an early point in its life cycle, there has been consistent, substantive, and rapid progress in making it practical from a performance standpoint. In this paper, we have shown that homomorphic encryption has enormous potential for enabling new use cases such as privacy-conscious sharing of i2b2 aggregate-level data in the cloud.

The proposed model is generalizable and is not tied to i2b2 aggregate-level data. In future work, we will extend its functionalities so that also individual-level data from the i2b2 data model could be shared in the cloud in a privacy-preserving way thus making a further step towards the realization of personalized medicine to its full potential.

References

1. Bai G, Jiang J, Flasher R. Hospital risk of data breaches. JAMA Intern Med. 2017 Apr 3;
2. Liu V, Musen MA, Chou T. Data Breaches of Protected Health Information in the United States. Jama. 2015;313(14):1471.
3. Ponemon Institute. Third Annual Benchmark Study on Patient Privacy & Data Security Sponsored by ID Experts. 2016;(May):50.
4. Murphy SN, Weber G, Mendis M, Gainer V, Chueh HC, Churchill S, et al. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). J Am Med Informatics Assoc. 2010;17(2):124–30.
5. Athey BD, Braxenthaler M, Haas M, Guo Y. tranSMART: An Open Source and Community-Driven Informatics and Data Sharing Platform for Clinical and Translational Research. AMIA Jt Summits Transl Sci proceedings AMIA Jt Summits Transl Sci. 2013;2013:6–8.
6. GenomePrivacy.org - The Genome Privacy and Security Community. Available from: https://genomeprivacy.org/
7. McLaren PJ, Raisaro JL, Aouri M, Rotger M, Ayday E, Bartha I, et al. Privacy-preserving genomic testing in the clinic: a model using HIV treatment. Genet Med. 2016 Aug 14;18(8):814–22.

8. Raisaro JL, Tramèr F, Ji Z, Bu D, Zhao Y, Carey K, et al. Addressing Beacon re-identification attacks: quantification and mitigation of privacy risks. J Am Med Informatics Assoc. 2017;0(0):1–8.

9. Jagadeesh KA, Wu DJ, Birgmeier JA, Boneh D, Bejerano G. Deriving genomic diagnoses without revealing patient genomes - supp Materials. Science (80-). 2017;347(692).

10. Chen F, Wang S, Jiang X, Ding S, Lu Y, Kim J, et al. PRINCESS: Privacy-protecting Rare disease International Network Collaboration via Encryption through Software guard extensionS. Bioinformatics. 2016;1–8.

11. Murphy SN, Chueh HC. A security architecture for query tools used to access large biomedical databases. Proceedings AMIA Symp. 2002;552–6.

12. Bernstein DJ, Duif N, Lange T, Schwabe P, Yang B-Y. High-speed high-security signatures. J Cryptogr Eng. 2012 Sep 14;2(2):77–89.

13. Fleurence RL, Curtis LH, Califf RM, Platt R, Selby J V., Brown JS. Launching PCORnet, a national patient-centered clinical research network. J Am Med Informatics Assoc. 2014 Jul 1;21(4):578–82.

14. Amazon Web Services (AWS) - Cloud Computing Services. Available from: https://aws.amazon.com/

15. Google Cloud Computing, Hosting Services & APIs | Google Cloud Platform. Available from: https://cloud.google.com/

16. Koblitz N. Elliptic curve cryptosystems. Math Comput. 1987 Jan 1;48(177):203–203.

17. Dwork C, Roth A. The Algorithmic Foundations of Differential Privacy. Found Trends® Theor Comput Sci. 2013;9(3–4):211–407.

18. Dwork C. A Firm Foundation for Private Data Analysis. Commun ACM. 2011 Jan;54(1):86–95.

19. DEDIS/EPFL. The Cothority Network Library - ONetNo Title.

20. Froelicher D, Egger P, Sousa JS, Raisaro JL, Huang Z, Mouchet C, et al. UnLynx: A Decentralized System for Privacy-Conscious Data Sharing. Proc Priv Enhancing Technol. 2017(4):152–70.

21. Bogdanov D, Bogdanov D, Laur S, Willemson J. Sharemind: a framework for fast privacy-preserving computations. Proc 13TH Eur Symp Res Comput Secur ESORICS 2008, LNCS. 2008;192--206.

22. Fan J, Vercauteren F. Somewhat Practical Fully Homomorphic Encryption. Cryptol ePrint Arch. 2012;

23. Yao AC. Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982). IEEE; 1982. p. 160–4.

24. Shamir A, Adi. How to share a secret. Commun ACM. 1979 Nov 1;22(11):612–3.

25. Camenisch J;, Stadler M, Camenisch J. Proof systems for general statements about discrete logarithms. 1997;