DOUBLE STANDARD SECURITY IN THE MANAGEMENT CLOUD BASED MOBILE APPLICATION

N.Suresh[1]

[1]*Student, CSE Department, Prakasam Engineering College, A.P., India,* ***surispn@gmail.com***

Abstract

Cloud-assisted mobile health (mHealth) monitoring, which applies the prevailing mobile communications and cloud computing echnologies to provide feedback decision support, has been considered as a revolutionary approach to improving the quality of healthcare service while lowering the healthcare cost. Unfortunately, it also poses a serious risk on both clients' privacy and intellectual property of monitoring service providers, which could deter the wide adoption of mHealth technology. This paper is to address this important problem and design a cloud- assisted privacy preserving mobile health monitoring system to protect the privacy of the involved parties and their data. Moreover, the outsourcing decryption technique and a newly- proposed key private proxy re-encryption are adapted to shift the computational complexity of the involved parties to the cloud without compromising clients' privacy and service providers' intellectual property. Finally, our security and performance analysis demonstrates the effectiveness of our proposed design.

Index Terms: *Mobile health (mHealth), Healthcare, Privacy, Outsourcing decryption, Key private proxy re-encryption..*

---------- *** ----------

1. INTRODUCTION

The development of cloud computing services is up the rate in which the organizations outsource their computational services or sell their idle computational speeding resources. Even though migrating to the cloud remains a tempting trend from a financial perspective, there are several other aspects that must be taken into account by companies before they decide to do so. One of the most important aspect refers to security: while some cloud computing security issues are inherited from the solutions adopted to create such services, many new security questions that are particular to these solutions also arise, including those related to how the services are organized and which kind of service/data can be placed in the cloud. Aiming to give a better understanding of this complex scenario, in this article we identify and classify the main security concerns and solutions in cloud computing, and propose taxonomy of security in cloud computing, giving an overview of the current status of security in this emerging technology.

Although the existing privacy laws such as HIPAA (Health Insurance Portability and Accountability Act) provide baseline protection for personal health record, they are generally considered not applicable or transferable to cloud computing environments [6]. Besides, the current law is more focused on protection against adversarial intrusions while there is little effort on protecting clients from business collecting private information. Meanwhile, many companies have significant commercial interests in collecting clients' private health data [7] and sharing them with either insurance companies, research institutions or even the government agencies. It has also been indicated [8] that privacy law could not really exert any real protection on clients' data privacy unless there is an effective mechanism to enforce restrictions on the activities of healthcare service providers.

Another major problem in addressing security and privacy is the computational workload involved with the cryptographic techniques. With the presence of cloud computing facilities, it will be wise to shift intensive computations to cloud servers from resource-constrained mobile devices. However, how to achieve this effectively without compromising privacy and security become a great challenge, which should be carefully investigated.

As an important remark, our design here mainly focuses on insider attacks, which could be launched by either malicious or non-malicious insiders. For instance, the insiders could be disgruntled employees or healthcare workers who enter the healthcare business for criminal purpose. It was reported that 32% of medical data breaches in medical establishments between January 2007 and June 2009 were due to insider attacks, and the incident rate of insider attacks is rapidly increasing. The insider attacks have cost the victimized institutions much more than what outsider attacks have caused. Furthermore, insider attackers are generally much harder to deal with because they are generally sophisticated professionals or even criminal rings who are adept at escaping intrusion detection. On the other hand, while outsider attacks could be trivially prevented by directly adopting cryptographic mechanisms such as encryption, it is non-trivial to design a privacy preserving mechanism against the insider attacks because we have to balance the privacy constraints and maintenance of normal operations of mHealthsystems. The problem becomes especially trickier for cloudassisted mHealth systems because we need not only to guarantee the privacy of clients' input health data, but also that of the output decision results from both cloud servers and healthcare service providers (which will be referred to as *the company* in the subsequent development).

In this paper, we design a cloud-assisted mHealth monitoring system (CAM). We first identify the design problems on privacy preservation and then provide our solutions. To ease the understanding, we start with the basic scheme so that we can identify the possible privacy breaches. We then provide an improved scheme by addressing the identified privacy problems. The resulting improved scheme allows the mHealth service provider (the company) to be offline after the setup stage and enables it to deliver its data or programs to the cloud securely. To reduce clients' decryption complexity, we incorporate the recently proposed outsourcing decryption technique into the underlying multi-dimensional range queries system to shift clients' computational complexity to the cloud without revealing any information on either clients' query input or the decrypted decision to the cloud. To relieve the computational complexity on the company's side, which is proportional to the number of clients, we propose a further improvement, leading to our final scheme. It is based on a new variant of key private proxy re-encryption scheme, in which the company only needs to accomplish encryption once at the setup phase while shifting the rest computational tasks to the cloud without compromising privacy, further reducing the computational and communication burden on clients and the cloud.

2. SYSTEM MODEL AND ADVERSARIAL MODEL

To facilitate our discussion, we first elaborate our cloudassisted mHealth monitoring system (CAM). CAM consists of four parties: the cloud server (simply the cloud), the company who provides the mHealth monitoring service (i.e., the healthcare service provider), the individual clients (simply clients), and a semi-trusted authority (TA). The company stores its encrypted monitoring data or program in the cloud server. Individual clients collect their medical data and store them in their mobile devices, which then transform the data into attribute vectors. The attribute vectors are delivered as inputs to the monitoring program in the cloud server through a mobile (or smart) device. A semi-trusted authority is responsible for distributing private keys to the individual clients and collecting the service fee from the clients according to a certain business model such as pay-as-you-go business model. The TA can

be considered as a collaborator or a management agent for a company (or several companies) and thus shares certain level of mutual interest with the company. However, the company and TA could collude to obtain private health data from client input vectors. We assume a neutral cloud server, which means it neither colludes with the company nor a client to attack the other side. This is a reasonable model since it would be in the best business interest of the cloud not to be biased. We admit that it remains possible for the cloud to collude with other malicious entities in our CAM, and we leave the CAM design under these stronger models as future work. We also do not assume that an individual client colludes with other clients. Our security model does not consider the possible side-channel attack due to the co-residency on shared resources either because it could be mitigated with either system level protection or leakage resilient cryptography. CAM assumes an honest but curious model, which implies all parties should follow the prescribed actions and cannot be arbitrarily malicious.

In the following, we briefly introduce the four major steps of CAM: Setup, Store, TokenGen and Query. We only illustrate the functionality of these components in this section while leaving the details in later sections.

At the system initialization, TA runs the Setup phase and publishes the system parameters. Then the company first expresses the flow chart of the mHealth monitoring program

as a branching program which is encrypted under the respective directed branching tree. Then the company delivers the resulting ciphertext and its company index to the cloud, which corresponds to the Store algorithm in the context. When a client wishes to query the cloud for a certain mHealth monitoring program, the *i*-th client and TA run the TokenGen algorithm. The client sends the company index to TA, and then inputs its private query (which is the attribute vector representing the collected health data) and TA inputs the master secret to the algorithm. The client obtains the token corresponding to its query input while TA gets no useful information on the individual query.

During the last phase, the client delivers the token for its query to the cloud, which runs the Query phase. The cloud completes the major computationally intensive task for the client's decryption and returns the partially decrypted ciphertext to the client. The client then completes the remaining decryption task after receiving the partially decrypted ciphertext and obtains its decryption result, which corresponds to the decision from the monitoring program on the clients' input. The cloud obtains no useful information on either the client's private query input or decryption result after running the Query phase. Here, we distinguish the query input privacy breach in terms of what can be inferred from the computational or communication information. CAM can prevent the cloud from deducing useful information from the client's query input or output corresponding to the received information from the client. However, the cloud might still be able to deduce side information on the client's private query input by observing the client's access pattern. This issue could be resolved by oblivious RAM technique [29], but this is out of the scope of this paper.

3. CAM DESIGN

To illustrate the fundamental idea behind this design, we start with the basic scheme, and then demonstrate how improvements can be made step-by-step to meet our design goal. Some of the variables in the following illustration may have already been defined in the previous sections. The system time is divided into multiple time periods, called *slots*, each of which can last a week or a month depending on specific application scenarios. There is an estimated maximum number of users N requesting access to the monitoring program in any given slot. When a client attempts to access the program, it is assigned an index *i in*[1,N] by TA.

3.1. Basic CAM

The following basic scheme runs the BF-IBE system as a sub-routine and is the fundamental building block in our overall design.

Setup: This algorithm is performed by TA, which publishes the system parameters for the BF-IBE scheme.
Store: This algorithm is performed by the company. For each node p_j whose child nodes are not leaf nodes, the company runs $C_{L(j)}$ = AnonEnc(id, PP,$L(j)$) and $C_{R(j)}$ = AnonEnc(id, PP,$R(j)$) to encrypt the child node indices under id with either id in $S_{[0;tj\ ]}$ or id in $S_{[tj+1;Max]}$, respectively. When the child nodes of p_j are leaf nodes, the company generates the ciphertext as $C_{L(j)}$ = AnonEnc(id, PP,$m_{L(j)}$) and $C_{R(j)}$ = AnonEnc(id, PP,$m_{R(j)}$), where $mL(j)$ and $mR(j)$ denote the attached information at the two leaf nodes, respectively. All the generated ciphertexts are delivered and stored in the cloud.

TokenGen: To generate the private key for the attribute vector v=(v_1,…, v_n), a client first computes the identity representation set of each element in v and delivers all the n identity representation sets to TA. Then TA runs the AnonExtract(id, msk) on each identity id in S_{vi} in the identity set and delivers all the respective private keys sk_{vi} to the client.

Query: A client delivers the private key sets obtained from the TokenGen algorithm to the cloud, which runs the AnonDecryption algorithm on the ciphertext generated in the Store algorithm. Starting from p1, the decryption result determines which ciphertext should be decrypted next. For instance, if $v1$ in [0, t_1], then the decryption result indicates the next node index $L(i)$. The cloud will then use $sk_{v(L(i))}$ to decrypt the subsequent ciphertext $C_{L(i)}$. Continue this process iteratively until it reaches a leaf node and decrypt the respective attached information.

3.2. CAM with Full Privacy Preservation

The basic scheme has the following security weakness: first, the identity representation set for a client's attribute vector v is known to TA, and hence TA can easily infer all the client's private attribute vector. Second, the client cannot protect his privacy from the cloud either because the cloud can easily find out the identity representation for the private key sk_{vi} , i *in* [1, n] by running identity test in MDRQs. The cloud can simply encrypt a random message under any attribute value v' until when it can use sk_{vi} to successfully decrypt the ciphertext, which means there is a match between $v' = vi$ and hence it successfully finds out vi. Third, neither can the data privacy of the company be guaranteed since the identity representation of the respective range is revealed to the cloud whenever the decryption is successful due to the match revealing property of MDRQs. The cloud can finally figure out most of the company's branching program since it has the private keys of all the system users.

To rectify the weakness of the basic scheme, we provide the following improvement. The high level idea (as shown in Fig.

1) is as follows: in order to avoid leaking the attribute vector to TA, the client obliviously submits his attribute vectors to TA so that he can obtain the respective private keys without letting TA get any useful information on his private vector. The client runs the outsourcing decryption of MDRQs to ensure the cloud completes the major workload while obtaining no useful information on his private keys. On the other hand, the company will permute and randomize its data using homomorphic encryption and MDRQs so that neither the cloud nor a client can get any useful information on its private information on branching program after a single query. Meanwhile, the company is also required to include the randomness in the randomization step in the encryption sent to TA to guarantee that TA can successfully generate tokens for clients.

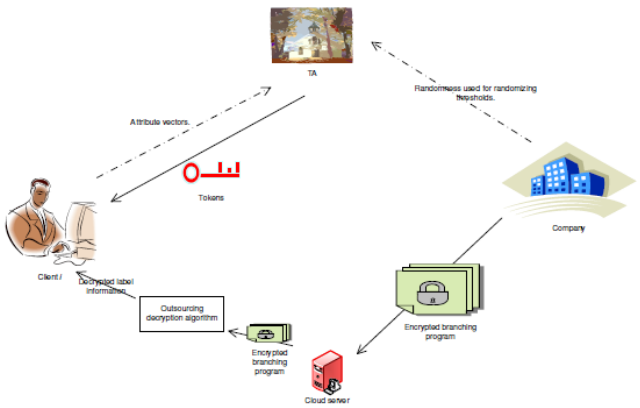

Fig. 1. CAM with Full Privacy Preservation

The improvement consists of four steps just as in the basic scheme. We will show how this improvement meets the desired security requirements.

Setup: This algorithm is performed by TA, which publishes the public parameter *PP* for the anonymous IBE.

Store: This algorithm is performed by the company. Let PRF(*s*, *i*) be a pseudo-random function which takes as input a secret key *s* and an *i*.
The company delivers all the ciphertexts, including the public key and symmetric key ciphertexts according to the permuted order, to the cloud while delivering both the pseudorandom function PRF(*s*, *i*), the random permutation function Q_i and the concerned attributes of the program, i.e., $\{a_1, \cdots, a_k\}$, to TA.

TokenGen: To generate the private keys for the attribute vector v=(v_1, $\cdots$, v_n), the *i*-th client first generates a public private key pair for a homomorphic encryption scheme, HEnc(·), and sends the public key and HEnc(*vj*) to TA.

Query: Starting from p1, the cloud runs Transform(C_{id}, tk_{id}) where id in $S_{t1+\_\delta i1}$ or $S_{[t1+\delta i1+1;Max']}$ and delivers the transformed ciphertext C'_{id} back to the client. Then the client runs AnonMaskDecryption(C'_{id}, *z*) to obtain the index of the subsequent node, either $Q_i[L(j)]$ or $Q_i[R(j)]$ and the respective symmetric key $kQ_i[L(j)]$ or $kQ_i[R(j)]$, depending on which range *v*1 falls in. He can then use the symmetric key to decrypt the underlying ciphertext, either $TC_{Qi[L(1)]}$ or $TC_{Qi[R(1)]}$, which will then be returned to the cloud with the respective index $Q_i[L(1)]$ or $Q_i[R(1)]$. The cloud continues to transform the subsequent ciphertext using the transformation key according to the returned index from the client. We note that the transformation key used by the cloud and the returned ciphertext correspond to an identical index since they are both permuted by an identical permutation function Q_i. They continue this process until the client reaches a leaf node and decrypts the respective decision result at a leaf node. The cloud obtains no information on either the decryption result or the company branching program due to the mask privacy of the AnonMaskDecryption algorithm.

We observe that, comparing with the basic scheme, the cloud obtains no useful information on the company's branching program. Due to the usage of permutation function, or the respective randomized thresholds from the pseudo-random function, and the security of the MDRQs system, the cloud obtains no useful information on the order of those intermediate nodes either. The cloud cannot find out the query vector v by performing identity test either because the transformation keys the cloud obtains during the query process cannot be used for identity testing. Indeed, those transformation keys leak no private information on the query vector v due to the mask privacy. The company can protect the data privacy from individual clients, especially the thresholds and orders of those branching nodes irrelevant to the client's final decision result, because the client does not even have a chance to perform the respective queries due to the semantic security of MDRQs and symmetric key encryption scheme. However, the client might be able to figure out the attribute thresholds of the intermediate nodes and their respective orders if those nodes lead to the final decision result due to the match revealing property of MDRQs, but this is all the possible side information the client can get. An interesting bonus of this improvement is that TA does not obtain much information on the company's branching program either. As a matter of fact, the only private information TA can infer from the information delivered by the company is the indices of the concerned nodes in the branching program.

3.3. Final CAM with Full Privacy and High Efficiency

Although the above improved scheme does meet the desired security requirements, the company may need to compute all the ciphertexts for each of N clients, which implies huge

computational overheads and may not be economically feasible for small mHealth companies. In this section, we provide a further improvement to reduce both the computational burden on the company and the communication overhead for the cloud. The high level idea (as shown in Fig. 2) is as follows. We employ a newly developed key private re-encryption scheme as an underlying tool. Instead of computing a ciphertext for each client, the company generates one single ciphertext, which will then be delivered to the cloud. The company will then obliviously deliver the identity threshold representation sets for the thresholds of the decisional branching nodes and the indexes of the concerned attributes to TA so that TA can generate the ReKeys corresponding to the rest clients in the system using the key private re-encryption scheme. The generated rekeys are then delivered to the cloud, which can then run the re-encryption scheme using the rekeys and the single ciphertext delivered by the company to generate

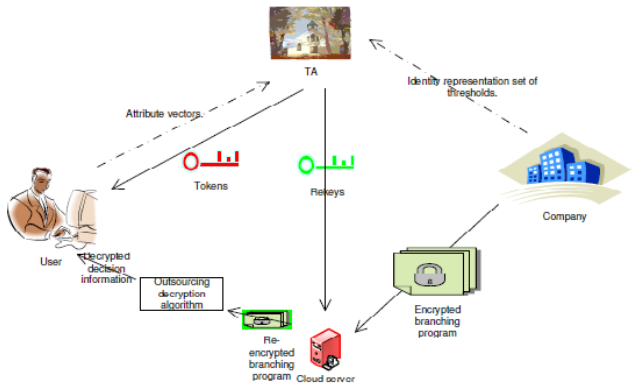

Fig. 2. Final CAM with Full Privacy and High Efficiency

the ciphertexts for the rest clients. The proposed re-encryption scheme incorporates the outsourcing decryption so that the other security and efficiency characteristics in the final CAM are inherited here. Besides, the decryption algorithm of the proxy re-encryption scheme induces much less interactions between clients and the cloud comparing with that in our improved scheme. Since the final scheme is based on the newly proposed key private proxy re-encryption scheme, we will present this scheme first.

4. CONCLUSION

In this paper, we design a cloud-assisted privacy preserving mobile health monitoring system, called CAM, which can effectively protect the privacy of clients and the intellectual proerty of mHealth service providers. To protect the clients' privacy, we apply the anonymous Boneh-Franklin dentitybased encryption (IBE) in medical diagnostic branching programs. To reduce the decryption complexity due to the use of IBE, we apply recently proposed decryption outsourcing with privacy protection to shift clients' pairing computation to the cloud server. To protect mHeath service providers' programs, we expand the branching program tree by using the random permutation and randomize the decision thresholds used at the decision branching nodes. Finally, to enable resource constrained small companies to participate in mHealth business, our CAM design helps them to shift the computational burden to the cloud by applying newly developed key private proxy re-encryption technique. Our CAM has been shown to achieve the design objective.

ACKNOWLEDGEMENT

We would like to express our sincere thanks to Sri. Dr. Kancharla Ramaiah Secretary and Correspondent, Prakasam Engineering College, Kandukur, A.P. India for his support with providing research environment . We are extremely thankful to our collogues , friends and family members who are cooperated in this work.

REFERENCES

[1] P. Mohan, D. Marin, S. Sultan, and A. Deen, "Medinet: personalizing the self-care process for patients with diabetes and cardiovascular disease using mobile telephony." *Conference Proceedings of the International Conference of IEEE Engineering in Medicine and Biology Society*, vol. 2008, no. 3, pp. 755–758. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19162765

[2] A. Tsanas, M. Little, P. McSharry, and L. Ramig, "Accurate telemonitoring of parkinson's disease progression by noninvasive speech tests," *Biomedical Engineering, IEEE Transactions on*, vol. 57, no. 4, pp. 884–893, 2010.

[3] G. Clifford and D. Clifton, "Wireless technology in disease management and medicine," *Annual Review of Medicine*, vol. 63, pp. 479–492, 2012.

[4] L. Ponemon Institute, "Americans' opinions on healthcare privacy, available: http://tinyurl.com/4atsdlj," 2010.

[5] A. V. Dhukaram, C. Baber, L. Elloumi, B.-J. van Beijnum, and P. D. Stefanis, "End-user perception towards pervasive cardiac healthcare services: Benefits, acceptance, adoption, risks, security, privacy and trust," in *PervasiveHealth*, 2011, pp. 478–484.

[6] M. Delgado, "The evolution of health care it: Are current u.s. privacy policies ready for the clouds?" in *SERVICES*, 2011, pp. 371–378.

[7] N. Singer, "When 2+ 2 equals a privacy question," *New York Times*, 2009.

[8] E. B. Fernandez, "Security in data intensive computing systems," in *Handbook of Data Intensive Computing*, 2011, pp. 447–466.

[9] A. Narayanan and V. Shmatikov, "Myths and fallacies of personally identifiable information," *Communications of the ACM*, vol. 53, no. 6, pp. 24–26, 2010.

[10] P. Baldi, R. Baronio, E. D. Cristofaro, P. Gasti, and G. Tsudik, "Countering gattaca: efficient and secure testing of fully-sequenced human genomes," in *ACM Conference on Computer and Communications Security*, 2011, pp. 691–702.

[11] A. Cavoukian, A. Fisher, S. Killen, and D. Hoffman, "Remote home health care technologies: how to ensure privacy? build it in: Privacy by design," *Identity in the Information Society*, vol. 3, no. 2, pp. 363–378, 2010.

[12] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE, 2008, pp. 111–125.

[13] "De-anonymizing social networks," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2009, pp. 173–187.

[14] I. Neamatullah, M. Douglass, L. Lehman, A. Reisner, M. Villarroel, W. Long, P. Szolovits, G. Moody, R. Mark, and G. Clifford, "Automated de-identification of free-text medical records," *BMC medical informatics and decision making*, vol. 8, no. 1, p. 32, 2008.

[15] S. Al-Fedaghi and A. Al-Azmi, "Experimentation with personal identifiable information," *Intelligent Information Management*, vol. 4, no. 4, pp. 123–133, 2012.

[16] J. Domingo-Ferrer, "A three-dimensional conceptual framework for database privacy," *Secure Data Management*, pp. 193–202, 2007.