Private Verification of Access on Medical Data: An Initial Study

Thaís Bardini Idalino[1], Dayana Spagnuelo[2(✉)], and Jean Everson Martina[3]

[1] School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada
[2] Interdisciplinary Centre for Security Reliability and Trust (SnT), University of Luxembourg, Luxembourg, Luxembourg
dayana.spagnuelo@uni.lu
[3] Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, Brazil

Abstract. Patient-centered medical systems promote empowerment of patients, who can decide on the accesses and usage of their personal data. To inspire a sense of trust and encourage the adoption of such systems, it is desired to allow one to verify whether the system has acted in accordance with the patients' preferences. However, it is argued that even audit logs and usage policies, normally used when verifying such property, may already be enough for one to learn sensitive information, e.g., the medical specialists a given patient has visited in the past. This is not only damaging for the patients, but is also against the interests of the medical system, which may lose back the trust earned and gain a bad reputation. Verifiability should not come at the expense of patients' privacy. It is, therefore, imperative that these systems take necessary precautions towards patient's information when providing means for verifiability. In this work we study how to realize that. In particular, we explore how searchable encryption techniques could be applied to allow the verification of systems in a private fashion, providing no information on patient's sensitive data.

Keywords: Verifiability · Audit · Compliance · Privacy · Searchable encryption · Patient-centered medical systems

1 Introduction

Verification is, by the pure meaning of the word, "the process of establishing the truth, accuracy, or validity of something"[1]. Verifiability is regarded in literature as a property desired in many information systems (e.g., [11,12,25]). It is also presented as one of the properties composing the principle of transparency, which is said to promote accountability and to realize people's right to privacy [36].

Verifiability has been studied as a mean for compliance with data access and usage policies [31]. It is presented from two perspectives: preventative and

[1] Definition taken from the Online Oxford Dictionaries.

J. Garcia-Alfaro et al. (Eds.): DPM/CBT 2017, LNCS 10436, pp. 86–103, 2017.
DOI: 10.1007/978-3-319-67816-0_6

detective. The preventative ensures that policies are enforced in IT operations. Thus non-compliant actions are prevented from even happening. The detective approach focus in validating the actions a posteriori.

The two approaches are not conflicting. In fact both can be combined in a way that the detective approach gives evidences that preventative techniques are in place and working properly. In some cases the combination of the two approaches is even mandatory. For example, medical systems usually need to be flexible and allow for emergency exceptions, such as break-the-glass. In these systems the preventative approach alone cannot enforce compliance. Detective approaches need to be in place in order to verify for obligations after the exceptional accesses happened.

In the medical systems domain, verifiability (also called auditability) has been explored with regard to access control (e.g., [14,24]). Even though there are solutions proposed for verifying access of personal data in medical systems (e.g., [20,21,32,33]), to the best of our knowledge, none do that while ensuring the details about patient's information are kept confidential [20,37]. In fact, according to Butin and Le Métayer [7], this is the most commonly used argument against verifiability in the context of personal data protection.

Allowing patients to manually verify compliance with policies is possible, but is not ideal. It would overwhelm them with the technical charge. A good verifiability solution in the medical domain should be *automatically executed*. To demonstrate good faith and commitment towards the fair use of personal data, it is desired that medical systems allow the verification process to be executed *independently*. In a way that the patient can choose to trust the system with the verification task, or to execute it with an external auditing tool. Moreover, it is imperative that the verifying solution ensures the *privacy of the subjects involved*. No personal and private information should be leaked during the verification process, even if unintentionally. However, those requirements are not easily achieved together. Commonly, verifiability solutions imply in the disclosure of information while privacy advocates the opposite. The independent verification requirement, while fostering the trust on the system may also become a privacy vulnerability if proper measures are not in place.

In this work we demonstrate how independent verifiability can be realized in a private fashion. We model an initial theoretical solution for detective compliance through verifiability in a patient-centered medical system. We use searchable encryption techniques for that. Our scheme allows for the access logs from medical system to be independently checked by a third party tool without leaking private information. It also protects the verification conditions by encrypting the queries executed by this third-party. Moreover, empowering users with the ability of privately checking compliance with access policies, helps supporting the confidence these users have in the system.

In what follows we present the related works and review the literature on searchable encryption techniques in Sect. 2. In Sect. 3 we contextualize and present the basic concepts on medical systems, and in Sect. 4 we model the requirements and the entities involved in our scheme. Section 5 deepens into the

details of our scheme, while in Sect. 6 we present a high-level analysis of the complexity and security of our proposal. Finally, in Sect. 7 we conclude our work and present the future directions.

2 Related Works

A survey from Reuben et al. [31] classifies the existing automated audits for privacy compliance verification. They study several solutions and separate them according to their auditing goals. The authors highlight three main goals: 1. *audit for ex-post obligations* – which regards compliance with after-the-fact obligations that cannot be verified beforehand, such as mandatory deletion of data after a fixed amount of time; 2. *audit for permitted exceptions* – which includes exceptional actions that happen in case of emergency (break-the-glass policies); and 3. *audits for access legitimacy* – which intends to demonstrate compliance with the data owner's preferences.

Audits for ex-post obligations do not necessarily imply on disclosure of personal data. In fact Butin and Le Métayer [7] propose a formal framework for verifying compliance in a privacy friendly way. They check compliance with data protection policies based on logs free of any personal data. However, they are not able to demonstrate compliance with access policies. They only verify properties such as "delete requests are fulfilled before expiration of request fulfillment delay", and "no personal data should appear in an abstract state after its global deletion delay has expired".

Audits for permitted exceptions and for access legitimacy pose more challenge for the privacy of personal and sensitive data. In most of the cases they mandate the analysis of audit logs, which contain information on *who* accessed *what kind* of information from *whom* [20,37]. In this work we intend to demonstrate how one could conduct these kind of audits in a private manner. For this purpose we show a model to automatically verify the latter (access legitimacy). Our scheme is capable of identifying accesses that do not match the user's preferences. Which can be later manually investigated for permitted exceptions (break-the-glass policies, for example). Automatic verification of permitted exceptions in a private manner would require a more in-depth study that is outside of the scope of our work. It will be subject of our future work.

In [20] authors point out security and privacy issues involved in making access policies and audit logs available in medical domain. They advocate that policies and logs, even though not containing personal and sensitive information (only references to it), may be enough for revealing details that should be kept private. Someone in possession of such policies and logs can gain knowledge of what kind of treatment a patient has received in the past, or what types of medical data are available. Authors advocate that by properly controlling the access to policies and logs it is possible to solve this privacy issue. They propose an adapted Information Accountability Framework [15] in which only the patient (data owners), medical professionals and medical authorities (e.g., government agency conducting audits) can access the policies and logs with restrictions according to their roles.

However, this work is not suitable to be applied to our context. In [20] independent auditing processes, one of our goals, are not considered. Even if this work was adapted to allow independent verification, the principle of privacy would still not be realized. External entities would still have access to more information than necessary to the purpose of verification. Restricted access control when applied in an uncontrolled environment (possibly insecure) does not suffice to prevent leakage of personal information.

While in [20] the confidentiality of sensitive information is realized only by controlling access to policies and logs, Walters et al. [37] propose a different solution for the problem: to operate on encrypted audit logs.

In [37], authors assume a scenario in which a system is being audited but the controllers of the system do not wish to share information from the audit logs with other entities. Similarly, the authors also believe it is possible to learn sensitive details about the system and the users by analyzing the logs. For example, one can instantly learn what actions were conducted by a given user. The authors build an scheme for conducting searches in encrypted audit logs. For each log registered, the system should define a few keywords with which this log can be found. It then distributes searching capabilities for those keywords only to specific authorized persons. Each log is encrypted with a key that can only be retrieved by persons that possess searching capabilities for, at least, one of its keywords. Consequently, this scheme only allows authorized persons to decrypt the audit logs.

The audit process presented in [37] cannot be fully independent though. It relies on the system providing searching capabilities to the auditor for the given set of actions he or she can audit. Despite that, this scheme is also not in accordance with the privacy principle. In our scenario, in order to verify compliance with the patient's preferences the auditor would search for log entries matching the set of allowed actions and be able to decrypt them, in detriment of the patient's privacy. Ideally the external entity should not be able to decrypt, only learning whether or not a given log entry matches a search (and consequently is an allowed action) would suffice.

There are several other works that, similarly to the one mentioned above, suggest schemes for privately processing personal data. The majority of those use searchable encryption techniques for that. In what follows we present the most relevant of those works while reviewing basic concepts of the technique.

2.1 Searchable Encryption

Searchable encryption (SE) techniques were initially introduced in the context of outsourced databases. With the growth of the amount of data generated, came an increasing need for outsourced options to store it. However, one cannot fully trust outsourced databases and may want to keep its data confidential. One possible solution to guarantee confidentiality involves encrypting the data before the storage on the database. Only the ones in possession of the key can decrypt it and learn its contents. However, denying the database access to the information increases the difficulty of performing queries and selectively retrieving data.

Searchable encryption techniques try to approach this problem by allowing the database to execute queries on encrypted data.

Search on encrypted data was initially introduced by Goldreich and Ostrovsky [18], and Song et al. [35]. It is, to this day, an active research area with three main research directions [5]: to improve efficiency; to improve security; and to enhance the expressiveness of the search. Usually we see a trade-off between them. For example, guaranteeing a stronger *security* usually compromises the *efficiency*.

An important scheme based on searchable encrypted index was first presented by Goh [17] and later considered in other works (i.e. [8,29,30] and many others). For each encrypted data, keywords are extracted and those are used to generate an encrypted index. In the outsourced database scenario, the indexes are generated by the client and sent with the respective encrypted data to the database. Later, the client can send an encrypted query and the indexes will help the database/server to search over the encrypted data without the need of decryption. Indexes and queries should not leak information about the encrypted data, while guaranteeing that clients obtain what they are searching for.

There are specific techniques for searching on public key [2,3,16] and symmetric key [10,17,35] encrypted data. The last one is known as searchable symmetric encryption (SSE). Several works presented solutions for searching single keywords [9,10,37]. Other schemes propose a search using more expressive keyword searches, such as conjunctions [4,6,19,34], ranges [4,34], or even dealing with keyword occurrence frequency [6]. This improves the expressiveness and security of searches, as opposed to perform several single-keyword searches and combining the results [29]. A few even more expressive schemes support general Boolean searches with conjunction, disjunction and negation of keywords in disjunctive normal form (DNF) and/or conjunctive normal form (CNF) [8,13,23,26,29,30]. We demonstrate later that these works are of a special interest since it is possible to model our problem into queries in a disjunctive normal form (DNF).

Symmetric searchable encryption are usually applied to scenarios where data owners want to query their own encrypted data stored in some third party server. In our work we propose the use of SSE techniques in a slightly different scenario, where data owners (patients) share their data with medical services and use SSE to independently verify accesses, while guaranteeing the confidentiality of their personal data.

It is necessary to note though, that we have a few different (and more relaxed) requirements in comparison to the conventional application of SSE in outsourced data storage. The first is related to the amount of data stored, searched and returned: while outsourced data applications may have to deal with large amounts of data, our application deals only with the event registers (logs) related to one specific patient (as shown in Sect. 3). We assume these logs to be in a smaller scale. This implies that the use of SSE algorithms with non-optimal search time is not prohibitive in our application. Second, the patient already has access to all encrypted data and uses the verifier only for auditing. Therefore, if the search returns all the data, which is an expected result for the cases

where no violation of the policy was made, we can save on communication and avoid returning everything again to the client, i.e. we can return just a positive message instead.

3 Technical Aspects of Medical Systems

The term "medical systems" is broad and encompasses several types of systems with different goals: clinical data management systems, telemedicine systems, hospital information systems, pharmaceutical, etc. In our work we only distinguish those which are patient-centered. The goal of these systems is to allow the patient to be in control of the personal data being processed. From this point on we refer to patient-centered systems simply as *medical systems*.

One example of a patient-centered medical system is Microsoft HealthVault[2], an online platform that allows users to gather, use and share health information. The information stored in the system can be provided manually by the user; or automatically by mobile applications or compatible medical devices. In this system the users are able to control which information is stored, deleted, and who will be able to access or edit their data [28]. Other example is the national Dossier de Soins Partagé[3] (Shared Care Dossier in English) from Luxembourg. In this system the goal is to facilitate the communication between health professionals intervening with a patient. Health data is uploaded to the system by authorized institutions, e.g., laboratories and hospitals, and shared with a default set of persons (the patient, the doctor assigned to him or her and the team related to them). But similarly to the Microsoft HealthVault, the patients have full control over sharing of data, being able even to revoke the default access privileges.

Generally speaking we can assume these patient-centered medical systems to adopt a discretionary access control system (DAC) [22]. In DAC systems the owner of a resource, in our case the patient, may grant or revoke access to other entities (users) based on their identities. We do not affirm that every patient-centered medical system implements DAC exactly as described in [22]. We just claim their access control method resembles DAC and could be modelled using it. For the sake of simplicity we assume discretionary access control policies as a set of fixed size clauses as shown in Eq. (1), where id_i is the identity of the person authorized to realize an action $action_j$ on the patient's data.

$$\pi = \{(id_i, action_j)\} \tag{1}$$

It is, however, unrealistic to assume one access control system to be the perfect fit for every variation of medical systems. We do not attempt doing that. We instead chose to model our solution based on DAC systems to demonstrate that private verification can be accomplished even in systems implementing highly malleable and granular access control mechanisms. We present arguments to

[2] https://www.healthvault.com/.
[3] https://www.esante.lu/portal/fr/espace-patient/le-dsp-au-quotidien,199.html?

endorse this claim in Sect. 5. And later, in Sect. 7, we discuss how our solution can also handle other types of policies richer in attributes.

Our simplified policy is only suitable to represent patient-centered medical systems though. In general these systems do not handle the definition of pre-conditions, post-conditions, obligations and other more complex policies that may be found in other types of medical system. To add more representativeness to the verification one could also explore revocation of access rights, which would mandate clauses to be time anchored. However, this is out of our scope. We restrict ourselves to the study of static policies and verification without temporal aspects.

Every action a person realizes on the patient's data, whether authorized or not, should be registered as an event in the audit logs. Similarly to how we defined the policies, we do for the register of events. We do not go into details on how they are in fact implemented because that may vary in different implementations. But according to a recent work [38] which surveys log files in the medical domain, it is reasonable to assume at least the following attributes would have to be registered in order to provide verifiability: 1. event identification ($action$) – the action performed; 2. date and time (t); 3. actor identification (id) – who performed the action; 4. object identification (ob) – the data that suffered the actions. Some standards are more complete and consider more attributes (i.e., RFC 3881 [27]), but in general these four attributes are commonly observed in medical systems [38]. We assume the register of events simply as the set of logs as displayed in Eq. (2).

$$L = \{(action, t, id, ob)\} \tag{2}$$

4 Model Description

As described in Sect. 3, our scenario assumes a patient-centered medical system. Users of such a system should be able to verify whether their data has been accessed in compliance to the access policy. We assume three different players:

- **Medical System:** Stores patient's data, which can be accessed by its owner (the patient), and few predetermined professionals. This decision is agreed with the patient through an access policy.
- **Patient:** May want to verify if specific statements of the policy are being enforced, or search for possible violations.
- **Verifier:** Third party tool or mechanism responsible for verifying compliance of the medical system with regard to specific statements of the agreed policy.

Additionally, we also assume the ideal solution would take into consideration the following requirements:

- **Automated verification:** The medical system should provide means for the patients to avoid the overburden of manually verifying logs;
- **Independent audit:** Allowing a third party to verify compliance with privacy policies demonstrates good faith and commitment towards the fair use of personal data;

– **Privacy:** During the auditing, patients' privacy should be ensured – only the strictly necessary information to determine compliance should be disclosed. From this information one should not be able to infer any personal details about the patients.

Patients should access and be able to export logs of actions performed on their data. However, data and the logs are private and should only be accessed by its owner (the patient) and a few designated medical staff. Therefore, both patient and the medical system are interested on keeping communications confidential. We chose to encrypt data with a symmetric key that is only known by the medical system and the patient. Keys differ for each patient of the system.

Patients may require the logs related to their data to check if the agreed policy is being followed. They can decrypt all logs received and verify by themselves, or they have the option to execute this task with an independent verifier. For that, the patient simply redirects the encrypted logs to the verifier. Since the verifier does not have access to the key used for encryption, a (good) traditional symmetric encryption is enough to guarantee that this verifier will not learn any information about the events these logs represent. Finally, in order to allow the verifier to operate over the encrypted logs while protecting the patient's privacy, we propose the use of symmetric searchable encryption (SSE).

4.1 Trust Model

The medical system we model is assumed to be honest, but not trustworthy. In systems that implement break-the-glass, for example, the policy may be relaxed and this can cause abuses. It may also be the case that the access control mechanism implemented does not flawlessly represent the policy agreed prior the disclosure of data. In both cases the medical system does not act ill-intentioned, but the patients' data can still be misused, and this may cause mistrust. Hence, the medical system's goal is to regain the trust of its users. This is realized by allowing them to independently verify whether the system has acted in compliance with the agreed policy. By doing that, we also avoid requiring the patient to place major trust in one single entity.

Our attacker model also assumes an honest-but-curious verifier, which will not actively behave dishonestly, but my retain any information disclosed to it. We also assume an external attacker, who will try to extract or infer information on the patients. The attacker is assumed to have access to the verifier, and any information exchanged between the other players. Because our goal is to demonstrate how independent verifiability can be achieved in a private manner (without leaking any sensitive information), we are only interested in what an attacker can learn through the use of the verifier. The capabilities of the attacker towards the medical system are not explored in this work.

In order to avoid a possible collusion between medical system and verifier, we suggest the implementation of several verifiers by different entities. In this way, the patients can double-check with different verifiers in case of suspicion. Verifiers would avoid collusion with medical systems in order to maintain reputation, and

medical systems would avoid collusion with verifiers as that can be identified by other verifiers. Verifiers can also be tested by the users with a set of logs and policies for which the expected results are known. Even though these approaches do not demonstrate the verifier correctness, they provide stronger evidences that can be used as criteria to support the choice of verifier.

It is important to note that we do not investigate into the matter of how to ensure the logs' accuracy and integrity. This topic is out of the scope of our work. We assume the medical system is honest and has its own reliable and trustworthy logging mechanism, and that it securely stores and handles data and logs. The following section presents in details our proposal for verification using searchable encryption.

5 Solving Verification with Searchable Encryption

Symmetric searchable encryption (SSE) schemes are popular in cloud settings. Data owners store encrypted data in an outsourced database, perform encrypted queries, and receive the encrypted data they searched for. We propose the use of SSE in a different setting: to verify whether the medical system is compliant to the access policy agreed with the patient. This verification is done through an external and independent audit. In our scenario, the verifier plays the role of the outsourced cloud service (even though it is not necessarily remote) and the patient is the data owner. We have added a third role played by the medical system, that is responsible for encrypting the data and generating the search indexes.

Next we present our scheme dividing it into the encryption of logs and index generation, the query generation and policy verification.

5.1 Encryption and Index Generation

For each patient that requires his or her logs, the medical system performs a *key agreement* process, where system and patient agree on a symmetric key k to be used for encryption and decryption. After that, the medical system encrypts each log individually and generates an index for each one of them, summarizing its content. The index includes all the *keywords* that can be searched in the encrypted data. Specifically for our scenario, the index of a log should contain the keywords related to the policy, such as the *action* registered by that log and the identity *id* of the user who performed the action (see Eq. (2)). The index generation depends on the SSE method used, but a common requirement is that no *keyword* in the index should be exposed. This is usually achieved by encrypting or through the use of scrambling-related techniques [6,13,17,29]. We abstract the process of encryption and index generation in Algorithm 1.

Algorithm 1. EncryptLogs($logs[n]$)

Input: array of $logs$ with n entries
for each $i \in \{1, \ldots, n\}$ **do**
  c[i] = Enc($k, logs[i]$)
  keywords = extractKeywords($logs[i]$)
  index[i] = generateIndex(k, keywords)
end for
Output: c, index

5.2 Query Generation

The medical system sends indexes and encrypted logs to the patient, who can redirect this information to the verifier for auditing purposes. On the patient's side, the main computation is related to the query generation. The query indicates which clauses the patient wants to verify. By generating one query containing each clause in the policy π (see Eq. (1)), it is possible to determine compliance. We present here Algorithm 2 as a generic algorithm for query generation.

In a traditional SSE approach, the data owner would generate an encrypted query and the database should simply execute this query over the indexes and return the respective encrypted data that matches the search. We adopt a similar approach. Here we assume that the patient has access to the policy agreed with the medical system. The patient then generates an encrypted query that contain the clauses from the policy, in order to check if the actions registered in the logs comply with the policy agreed. If the policy and the key used to encrypt the logs and indexes do not change, this query could also be further reused by the same patient.

We represent our query as a Boolean expression in a disjunctive normal form (DNF). We assume a simple policy containing only a set of s *identities* and t *actions* as *keywords*, and relations in the format (id_i, $action_j$) representing a clause of the policy allowing a person identified by id_i to execute $action_j$. To search for all logs that match the policy, the patient can generate a query in the DNF form as follows: $(id_{i_1} \wedge action_{j_1}) \vee \ldots \vee (id_{i_s} \wedge action_{j_t})$, where $(id_i \wedge action_j)$ is an allowed relation of the policy.

We abstract the query generation with a call to *"generate query"*. The details of this generation depend on the SSE method, but it basically identifies the clauses from the DNF expression and perform specific computations depending on the method used. Since we don't want the verifier to obtain information about the encrypted logs, some computation must be performed on the query as well to guarantee its confidentiality. This is a reasonable assumption considering that several SSE algorithms already guarantee that by using encryption or scrambling-based techniques [8,23,29]. As an example, the query generation by Moataz and Shikfa [29] consists on converting keywords to vectors, and applying consecutive multiplications, sums and divisions to them. The confidentiality in this case is guaranteed by incorporating random integers in the query computation (see [29, Sect. 4.2] for more details).

Note that the query generation is quite flexible, since it allows the patient to search for a range of different options. For example, he or she can search for all logs that match the policy, for some combination of specific clauses from the policy, or even for logs that do not match the policy by simply negating the search expression. It is important to note that most SSE schemes (specially the most traditional ones) search for single keywords on encrypted data. Here we require the use of a more expressive SSE that supports Boolean queries, such as the solutions proposed in [8,23,29]. Algorithm 2 summarizes the process of query generation.

Algorithm 2. GenerateQuery(π)

Input: Policy $\pi = \{(id_i, action_i)\}$ with m clauses
DNF = empty string
for each $i \in \{1, \ldots, m\}$ **do**
  DNF = DNF $\|(id_i \wedge action_i)$
  if $i \neq m$ **then**
    DNF = DNF $\|\vee$
  end if
end for
$\mathcal{Q}$ = generateQuery(DNF)
Output: $\mathcal{Q}$

Recall that we created indexes for the encrypted logs with the keywords (id_i, $action_j$) contained in each log. The verifier can then search through the logs indexes and identify the ones that match at least one of the conjunctions ($id_i \wedge action_j$) from the query. Here we considered small policy clauses, but if necessary, we can easily adapt the query to be more expressive. For example, by considering extra information such as identification of the objects that suffered the actions. In order to incorporate extra keywords in the search, these keywords also need to be incorporated in the policy and during the generation of the logs' indexes.

5.3 Policy Verification

After the verifier receives the encrypted logs, respective indexes, and the query $\mathcal{Q}$, it has enough information to perform the verification for policy compliance. The search process consists on going through the encrypted logs to find the ones that match the query. For each log, the verifier obtains the corresponding index and use it to check if it matches the query. Here we call this comparison "test" and the logs that "pass" the test are added to the vector of results. If a log pass a test, it means that this log contains at least all the keywords from one of the conjunctions of the query, which represents compliance with one of the clauses of the policy.

Note that the search depends on the SSE method as well. Some constructions propose visiting each encrypted data and its indexes [29], while others present

some more efficient search methods [13,23,30]. After the search, the verifier sends to the patient a list of encrypted logs that match the query. As logs, indexes, and queries are encrypted, the verifier is not able to learn anything about the confidential information. As an example, the verification by Moataz and Shikfa [29] consists on visiting every index and comparing it to the query. Generally speaking, every index is multiplied by the query and the ones that output a result equals to "1" correspond to logs that satisfy at least one clause from the policy. Algorithm 3 summarizes the our verification process.

Algorithm 3. Search$(\mathcal{Q}, c[n], index[n])$

Input: Encrypted query $\mathcal{Q}$, vector c with n encrypted logs, and their indexes
for each $i \in \{1, \ldots, n\}$ **do**
  $r = \text{test}(\mathcal{Q}, index[i])$
  if $r =$ **true then**
    $result$.add$(c[i])$
  end if
end for
Output: $result$

The results can be simplified by returning the number of logs that match the query, or a custom message for the special cases, such as "All logs match your query", or "No logs match your query". If further investigation is desired, Algorithm 3 can easily be adapted to return the logs that caused a mismatch. If the patient is interested in learning the cause for the mismatch he or she can decrypt those logs (using the key k) and understand what event is not compliant to the policy. Alternatively, these logs could be redirected to the medical system in order to inquire for a justification. How to better display and interpret the results, or how to request for justification are definitely relevant issues, but we understand they would require a research on their own. We refrain from delving into those matter in this work.

6 Complexity and Security Analysis

Several searchable encryption schemes are designed for the settings of big data applications, where there is a large amount of encrypted data and it is, for example, unfeasible to search through every single record. Our scenario is slightly different and some of the assumptions in those settings are not applicable here. In what follows we discuss how technical aspects of SSE methods impact our solution. We first examine aspects of computational complexity of those methods (Subsect. 6.1) and later we discuss about their security (Subsect. 6.2).

6.1 Complexity

The efficiency of our scheme is directly related to the efficiency of the SSE method used. However, the use of non-optimal SSE methods, such as [8,29],

while prohibitive for big data applications, is acceptable in our scenario. In fact, SSE techniques are very well suited for our application. Our verifier processes only the audit logs related to one specific patient. These logs are assumed to be small pieces of data and in a much smaller scale than in cloud settings. Consequently, the efficiency problems presented in outsourced databases are not applicable here. Moreover, the generation of keywords in our application does not require complex calculations. The keywords are defined by the policy and logs, and can be automatically extracted from those.

Other more efficient SSE methods, such as the ones with sub-linear search time [23,30], could also be considered here. In this case, the efficiency would depend on the query we are searching for. When searching for all logs that match the policy, it is expected the result to be close to the total amount of logs n. Hence, our search complexity will end up being close to $O(n)$ as well. However, when searching for the logs that *do not* match the policy, or that match some specific patters, we should expect a small number of results. In this case, methods that have search time close to the number of results may be the right choice. We assume the complexity of our worst-case scenario to be $O(n)$.

Intuitively one would tend to believe that the most efficient SSE methods on the literature are the best fit. However, there are trade-offs on these methods that need to be consider. Some of these methods use complicated structures and increase the spacial complexity, and others end up revealing parts of sensitive information. For a more extensive discussion on the trade-offs related to expressiveness of the query, efficiency, and security, the reader may want to refer to [5]. The choice of the method is not straightforward, it needs to be carefully studied. However, we suggest that in cases where the number of logs (n) is reasonable, it is a good practice to prioritize secure over efficient methods, even if they offer search complexity of $O(n)$.

6.2 Security

The confidentiality of personal information in our scheme is provided by the chosen symmetric key encryption algorithm. To avoid brute force and the most common attacks it is recommended to use encryption algorithms that have at least 112 bits of security (i.e. AES) [1]. Moreover, the use of deterministic encryption algorithms commonly implies in the leakage of patterns [5]. Therefore, the most secure SSE schemes are usually non-deterministic.

The security of the scheme is not only given by the encryption algorithm though, it also depends on the security of the SSE method itself. The SSE methods in the literature present a concern on the amount of information that can be inferred by the results of the search. Although they do not reveal directly the content of the encrypted data, the majority of these schemes will not prevent probabilistic analysis if the same data is repeatedly searched. This is a problem common to any application of this nature.

In our application the verifier returns all the results that match an specific query. This means that the verifier knows which encrypted logs are being

returned, but not their plain content. The same applies to the attacker we consider in our model, since we assume it has access to the verifier and any exchanged message. There are SSE methods that aim to hide all information. In this way, the verifier is not even able to detect which logs are being returned to the data owner. However, these SSE methods are usually based on oblivious RAMs (ORAMs), and are not efficient in practice (for more details, see survey [5]).

We understand that the searches will usually be related to the logs that match or do not match the policy. By analyzing the number of results from an specific query one could guess which search was performed. The search with several results is likely to be a search for all logs that match the policy, and the search with a few results is probably for the logs that do not match the policy. In this sense the verifier (or attacker) would be able to identify which logs match/do not match the policy, but would not be able to learn their content. We do not consider this as a threat to our scheme. A well chosen encryption algorithm would make sure that encrypted logs and queries are not available in plain text (they are encrypted or scrambled depending on the SSE scheme). Nonetheless, it is important to consider this case when applying our solution to other scenarios.

It is also important to note that any technical limitation of the underlying schemes (symmetric encryption or SSE) also reflects on a limitation of our proposed solution. We can cite, for example, the case of compromised or revoked private keys. In this case every data encrypted with that key is assumed to be compromised as well. A known solution to neutralize the potential damage is to reduce the lifetime of the key, and for example, use session keys instead of a single private key. In our scenario this solution would come at the cost of recalculating the queries, which could no longer be reused. This is the classical trade-off between efficiency and security.

Furthermore, a slight modification of our scheme is also needed to cope with the problem of compromised keys. Compromising one session key is enough for breaking privacy, even if forward and backwards secrecy are maintained, and no other message is obtained. The damage cause by compromising one session key is proportional to the amount of logs encrypted with that key. Therefore, to minimize this problem, only a small subset of logs should be verified at a time, i.e. the logs of the day or past week. Note that the maximum number of logs that are encrypted by the same session key is determined by the security guarantees required for specific applications of our scheme.

7 Discussion and Conclusion

In this work we modeled a scheme for verifying data access in the context of patient-centered medical systems. Our scheme is based on Symmetric Searchable Encryption methods and suggests how to meet the three requirements we deem imperative in the medical context: 1. *automated verification* – patients should not be required to manually verify audit logs; 2. *independent audit* – demonstrates the honest intentions of the medical system and helps building reputation; and 3. *privacy* – protects the right for privacy of the patients.

We propose the introduction of an entity for auditing the *Medical System* on behalf of the *Patient*. This entity is the *Verifier*. The implementation of the verifier as a Transparency Enhancing Tool controlled by the patient or by a third party would suffice to accomplish requirements for automated independent audit (requirements 1 and 2).

The privacy principle poses the biggest challenge as it conflicts with the other two requirements. To be able to verify whether a system acted in compliance with a given policy the verifier needs access to audit logs. However, those logs may reveal private information about the patients [20,37]. Revealing them is outside of the patient's interest. But not revealing them would mean the patient needs to trust the medical system with the verification. Obliging the patient to place major trust in one single entity. In this paper we demonstrate that Symmetric Searchable Encryption (SSE) can be adapted to provide the right balance between the requirements.

By using SSE methods we allow the verifier to operate on encrypted logs. In this way the interests of the patient are protected. Our scheme defines that compliance is achieved whenever a log entry matches, at least, one clause in the policy. This is possible since the policy describes every allowed action in the system. The SSE method allows then the verifier to search for those matches over encrypted audit logs and the obfuscated policy clauses. This ensures the verifier, and consequently the attacker, will not learn any sensitive information. The only information the one can learn is the number of logs that match (or not) a given search. This is, however, acceptable in our scheme since the logs are encrypted and indistinguishable from each other, and the policy is obfuscated. We understand that this is a necessary trade-off between privacy and verifiability in medical systems. Moreover, this does not violate requirement 3, which foresees a *minimal* disclosure to determine compliance.

One may question the simplicity of our policy model. Only the identity of the actor and the action executed are considered. This was deliberately done to simplify the description. Our model supports other types of access control policies and any number of attributes. This can be realized by computing extra keywords in the indexes and conjunctions of the query. The model we propose can handle the verification of actors' roles, attributes, sections of personal data, for example. Given the medical system supports such attributes in its policy and register them in the audit logs.

At this stage, our proposal is limited to the identification of log entries that match and the ones that do no match a given policy. The latter can then be manually investigated for permitted exceptions (break-the-glass policies), or other events also relevant in medical systems, such as delegation. Including these events in our verifier is a natural evolution of our proposal. As our next step we plan to investigate this matter.

We also foresee the formalization of our scheme as a future work. A good starting point is the work from Butin and Le Métayer [7]. We plan to extend that model now accounting for access legitimacy. To do so we will have to take a more careful look into the representation of events (logs) and authorized actions (policy), and to define protocols for obtaining and transferring these data between

the players. We will also need to investigate deeper on the SSE schemes in order to select the most suitable ones. The extension of searching capabilities is directly dependant on the evolution of these schemes.

Finally, another interesting evolution of our work is to allow the verification of data access even in the presence of a dishonest medical systems. A possible starting point is to study how to ensure the logs integrity and accuracy. This is important to prevent medical systems from intentionally removing or altering logs that do not comply with the policy.

Acknowledgments. Thais Bardini Idalino acknowledges funding granted from CNPq-Brazil [233697/2014-4]. Dayana Spagnuelo's research is supported by the Luxembourg National Research Fund (FNR), AFR project 7842804 - TYPAMED. The authors would also like to thank Dr. Gabriele Lenzini and Prof. Peter Y.A. Ryan for contributing to this work with relevant discussions and valuable advice.

References

1. Barker, E.: NIST Special Publication 800-57 Part 1 Revision 4—Recommendation for Key Management (Part 1: General) (2016)
2. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24676-3_30
3. Boneh, D., Raghunathan, A., Segev, G.: Function-private identity-based encryption: hiding the function in functional encryption. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 461–478. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40084-1_26
4. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007). doi:10.1007/978-3-540-70936-7_29
5. Bösch, C., Hartel, P., Jonker, W., Peter, A.: A survey of provably secure searchable encryption. ACM Comput. Surv. (CSUR) **47**(2), 18 (2015)
6. Bösch, C., Tang, Q., Hartel, P., Jonker, W.: Selective document retrieval from encrypted database. In: Gollmann, D., Freiling, F.C. (eds.) ISC 2012. LNCS, vol. 7483, pp. 224–241. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33383-5_14
7. Butin, D., Le Métayer, D.: Log analysis for data protection accountability. In: Jones, C., Pihlajasaari, P., Sun, J. (eds.) FM 2014. LNCS, vol. 8442, pp. 163–178. Springer, Cham (2014). doi:10.1007/978-3-319-06410-9_12
8. Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 353–373. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_20
9. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005). doi:10.1007/11496137_30
10. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. J. Comput. Secur. **19**(5), 895–934 (2011)

11. Dreier, J., Giustolisi, R., Kassem, A., Lafourcade, P., Lenzini, G.: A framework for analyzing verifiability in traditional and electronic exams. In: Lopez, J., Wu, Y. (eds.) ISPEC 2015. LNCS, vol. 9065, pp. 514–529. Springer, Cham (2015). doi:10.1007/978-3-319-17533-1_35
12. Dreier, J., Jonker, H., Lafourcade, P.: Defining verifiability in e-auction protocols. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, pp. 547–552. ACM (2013)
13. Fisch, B.A., Vo, B., Krell, F., Kumarasubramanian, A., Kolesnikov, V., Malkin, T., Bellovin, S.M.: Malicious-client security in blind seer: a scalable private DBMS. In: 2015 IEEE Symposium on Security and Privacy (SP), pp. 395–410. IEEE (2015)
14. Flores, A.E., Vergara, V.M.: Functionalities of open electronic health records system: a follow-up study. In: 6th International Conference on Biomedical Engineering and Informatics, pp. 602–607. IEEE (2013)
15. Gajanayake, R., Sahama, T.R., Lane, B., Grunwell, D.: Designing an information accountability framework for eHealth. In: IEEE Healthcom 2013 15th International Conference on E-Health Networking, Application and Services. Instituto Superior de Ciências Sociais e Políticas - Technical University of Lisbon, Lisbon, Portugal, June 2013. https://eprints.qut.edu.au/60690/
16. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009)
17. Goh, E.J., et al.: Secure indexes. IACR Cryptology ePrint Archive 2003, 216 (2003). http://eprint.iacr.org/2003/216
18. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. J. ACM (JACM) **43**(3), 431–473 (1996)
19. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24852-1_3
20. Grunwell, D., Gajanayake, R., Sahama, T.: The security and privacy of usage policies and provenance logs in an information accountability framework. In: Proceedings of the Eighth Australasian Workshop on Health Informatics and Knowledge Management (HIKM2015), vol. 164, pp. 33–40. Australian Computer Society (2015)
21. Haas, S., Wohlgemuth, S., Echizen, I., Sonehara, N., Müller, G.: Aspects of privacy for electronic health records. Int. J. Med. Inf. **80**(2), e26–e31 (2011)
22. Hu, V.C., Ferraiolo, D., Kuhn, D.R.: Assessment of access control systems. US Department of Commerce, National Institute of Standards and Technology (2006)
23. Kamara, S., Moataz, T.: Boolean searchable symmetric encryption with worst-case sub-linear complexity. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 94–124. Springer, Cham (2017). doi:10.1007/978-3-319-56617-7_4
24. King, J.T., Smith, B., Williams, L.: Modifying without a trace: general audit guidelines are inadequate for open-source electronic health record audit mechanisms. In: Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium, pp. 305–314. ACM (2012)
25. Kremer, S., Ryan, M., Smyth, B.: Election verifiability in electronic voting protocols. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 389–404. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15497-3_24
26. Kurosawa, K.: Garbled searchable symmetric encryption. In: Financial Cryptography, vol. 2014, pp. 234–251 (2014)

27. Marshall, G.: Security audit and access accountability message XML. Technical report, RFC 3881 (2004)
28. Microsoft: Microsoft Privacy Statement (2017). https://privacy.microsoft.com/en-gb/privacystatement. Accessed 15 May 2017
29. Moataz, T., Shikfa, A.: Boolean symmetric searchable encryption. In: Proceedings of the 8th ACM SIGSAC symposium on Information, Computer and Communications Security, pp. 265–276. ACM (2013)
30. Pappas, V., Krell, F., Vo, B., Kolesnikov, V., Malkin, T., Choi, S.G., George, W., Keromytis, A., Bellovin, S.: Blind seer: a scalable private DBMS. In: 2014 IEEE Symposium on Security and Privacy (SP), pp. 359–374. IEEE (2014)
31. Reuben, J., Martucci, L.A., Fischer-Hübner, S.: Automated log audits for privacy compliance validation: a literature survey. In: Aspinall, D., Camenisch, J., Hansen, M., Fischer-Hübner, S., Raab, C. (eds.) Privacy and Identity 2015. IAICT, vol. 476, pp. 312–326. Springer, Cham (2016). doi:10.1007/978-3-319-41763-9_21
32. Røstad, L.: An initial model and a discussion of access control in patient controlled health records. In: 2008 Third International Conference on Availability, Reliability and Security, ARES 2008, pp. 935–942. IEEE (2008)
33. Seneviratne, O., Kagal, L.: Enabling privacy through transparency. In: 2014 Twelfth Annual International Conference on Privacy, Security and Trust (PST), pp. 121–128. IEEE (2014)
34. Shi, E., Bethencourt, J., Chan, T.H., Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: 2007 IEEE Symposium on Security and Privacy, SP 2007, pp. 350–364. IEEE (2007)
35. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, S&P 2000, Proceedings, pp. 44–55. IEEE (2000)
36. Spagnuelo, D., Lenzini, G.: Transparent medical data systems. J. Med. Syst. **41**(1), 8 (2017)
37. Waters, B.R., Balfanz, D., Durfee, G., Smetters, D.K.: Building an encrypted and searchable audit log. NDSS **4**, 5–6 (2004)
38. Wickramage, C., Sahama, T., Fidge, C.: Anatomy of log files: implications for information accountability measures. In: 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom), pp. 1–6. IEEE (2016)