Annals of Telecommunications (2020) 75:103–119
https://doi.org/10.1007/s12243-020-00759-2

A break-glass protocol based on ciphertext-policy attribute-based encryption to access medical records in the cloud

Marcela T. de Oliveira[1] · Alexandros Bakas[2] · Eugene Frimpong[2] · Adrien E. D. Groot[1] · Henk A. Marquering[1] · Antonis Michalas[2] · Silvia D. Olabarriaga[1]

Received: 5 July 2019 / Accepted: 1 December 2019 / Published online: 7 March 2020

Abstract
In emergency care, fast and efficient treatment is vital. The availability of Electronic Medical Records (EMR) allows healthcare professionals to access a patient's data promptly, which facilitates the decision-making process and saves time by not repeating medical procedures. Unfortunately, the complete EMR of a patient is often not available during an emergency situation to all treatment teams. Cloud services emerge as a promising solution to this problem by allowing ubiquitous access to information. However, EMR storage and sharing through clouds raise several concerns about security and privacy. To this end, we propose a protocol through which all treatment teams involved in the emergency care can securely decrypt relevant data from the patient's EMR and add new information about the patient's status. Furthermore, our protocol ensures that treatment teams will only access the patient's EMR for the period during which the patient is under their care. Finally, we present a formal security analysis of our protocol and some initial experimental results.

Keywords Ciphertext-policy attribute-based encryption · e-health privacy · Access control · Electronic medical records · Emergency care · Secure cloud storage · Break-glass access

1 Introduction

Time is critical in emergency situations. In a short time frame, health professionals need to evaluate the patient's condition, decide upon the treatment, transport the patient to the adequate care centre, and perform the required intervention. The triage and diagnosis demand and generate a large amount of data, which needs to be shared between treatment teams along the whole process. The use of a single interoperable Electronic Medical Record (EMR) improves the overall quality of care [1], leading to a substantial reduction of unnecessary investigations and to an optimized communication among the healthcare professionals involved in the treatment.

The use of a cloud storage service allows practical and dynamic management of EMRs since a cloud infrastructure enables remote and ubiquitous access to data. However, one of the biggest concerns users have about cloud storage is data security. No one wants their sensitive data jeopardized. Recently, studies propose to send the EMR to a cloud service provider, where it is encrypted and stored. In this scenario, the

Marcela T. de Oliveira and Alexandros Bakas contributed equally to this work.

✉ Marcela T. de Oliveira
m.tuler@amsterdamumc.nl

✉ Alexandros Bakas
alexandros.bakas@tuni.fi

Eugene Frimpong
eugene.frimpong@tuni.fi

Adrien E. D. Groot
a.e.groot@amsterdamumc.nl

Henk A. Marquering
h.a.marquering@amsterdamumc.nl

Antonis Michalas
antonios.michalas@tuni.fi

Silvia D. Olabarriaga
s.d.olabarriaga@amsterdamumc.nl

[1] Amsterdam University Medical Centers, University of Amsterdam, Amsterdam, The Netherlands

[2] Tampere University, Tampere, Finland

key used for data encryption is known by the cloud provider, which does not protect the EMR against internal attacks [2]. Researchers suggest to encrypt the EMR with a secret key before storing it in the cloud [3, 4]. This means that the secret key needs to be pre-shared with all users that wish or need to access the EMR at any time throughout the treatment. Nevertheless, if a user needs to be revoked from the process of treatment, the EMR must be re-encrypted with a fresh key and the new key must be distributed to the other legitimate users. Therefore, revocation in this scenario is not efficient.

In the case of acute stroke care, the phrase 'Time is brain' conveys the idea that minutes can make the difference between life and death [5]. The availability of patient data is of paramount importance for the triage, diagnosis and treating centre selection. Therefore, it is necessary to provide access to patient data, even if the patient cannot consent explicitly, which is often the case in patients with acute stroke. The so-called break-glass access mechanism provides emergency access to the patient's EMR in such situations. Although some studies approach the break-glass access to encrypted EMR [6–8], its revocation after an emergency is still a problem. For security and privacy sake, immediately after the emergency situation ends, the access needs to be revoked. In addition, revoking a user's access must not affect the access of the rest of the users. Therefore, our goal is to provide a solution that allows break-glass access to a patient's EMR only during an emergency situation for only authorized treatment teams.

Our contribution We describe a protocol to provide access to a patient's encrypted EMR during acute stroke treatment with an additional security mechanism, which ensures authorization only for the period when the access is necessary. The protocol securely enables sharing of EMR among multiple treatment teams through a cloud platform. The proposed solution adopts the concept of attribute-based encryption (ABE) associated with policies defined for emergency situations. Additionally, it adopts token authentication to grant and revoke access during the timeline of acute stroke treatment. We prove the security of our scheme by constructing a simulator that is computationally indistinguishable from the real protocol. Moreover, we also prove the resilience of our scheme against a set of attacks defined in the threat model. Finally, we prove the effectiveness and robustness of our scheme in real-world situations by implementing the core functions of the proposed protocol.

Organization Section 2 discusses related works and Section 3 summarizes the flow of patient information during stroke emergency. Section 4 defines the cryptographic primitives used throughout the paper. In Section 5, we present the main entities that participate in our system model, and in Section 6, we define both the problem statement and the considered threat model. In Section 7, we describe our protocol, and in Section 8, we analyze its security against malicious behaviour. Section 9 presents the results of our experiments on the execution times of the proposed protocol's core functions. Section 10 discusses the results and limitations. Section 11 presents preliminary conclusions.

2 Related work

Break-glass is a term used to refer to security solutions that provide access to information in emergency situations.

In [9], the author proposed an encryption scheme for cloud storage that can be broken by any one exactly once, in a detectable way. The motivation for break-glass is the case when the legitimate user wants to decrypt the data previously uploaded to the cloud, but she lost all her secret keys. Our work, however, focuses on healthcare emergency situations, where the break-glass condition is valid to provide EMR availability to support triage, diagnosis and treatment. Very few research works have considered this requirement.

One of the earliest arguments for a break-glass concept for the healthcare case was formulated by Povey [10]. He stated that the basic approach of an optimistic security system is to assume that any emergency situation requesting data access is legitimate and should be granted. Petrisch and Bruker presented a generic break-glass model in [11] where the data subjects are allowed to override specific access control permissions. In [12], Zhang et al. proposed a concrete break-glass solution based on two-factor encryption: password-based encryption and master secret key based encryption. In [13] the authors presented 'Rampole', a model that implements access permissions in a fine-grained manner using a declarative query language to explicitly specify a break-glass decision procedure. None of the approaches described above supports attribute-based access control.

In [6], the authors use attribute-based encryption (ABE) techniques to control access to patient data. This study approaches break-glass access under emergency scenarios using a unique authority to authenticate the medical staff to access the data. To revoke access, the data needs to be re-encrypted with a new key. Brucker et al. [7] presented an integration of fine-grained break-glass concepts into a system based on ABE. The authors present multi-levelled break-glass access control; however, the solution does not enable revoking access after it is granted. Yang et al. [8] proposed a solution for ABE access control in which the patient pre-shares her password with the emergency contact person. When the patient reaches an emergency the situation, the contact person utilizes the password to derive the break-glass key and to

decrypt the patient's medical files. Even though [6–8] present interesting solutions for the break-glass situation, they do not provide a concrete and efficient solution for access revocation.

Back in 1999, in [14, 15], the authors approach the problem of key revocation in a dynamic group by proposing protocols for key management for multicasting. Similarly to our work, the authors were motivated about the case where a large number of people joining/leaving the authorization groups might affect the efficiency of the cryptographic scheme. Rafaeli and Hutchison presented a survey of key management for secure group communication [16]. Although the works in [14–16] present techniques to minimize the number of message transmissions required, their schemes still need to rekey the multicast authorized group after a revocation. Our approach overcomes the rekeying problem by using a ciphertext-policy ABE (CP-ABE) scheme and an access control token scheme to grant and revoke access dynamically without the need to re-encrypt the patient EMR. In addition, our protocol supports the involvement of multiple treatment teams, even from different institutions, which brings the solution closer to a real emergency scenario.

3 Patient data sharing during acute stroke emergency

Acute stroke care is a complex collaboration of various parties: professionals at the emergency call centre, ambulance nurse and driver, medical doctors and nurses at the hospital. Currently treatment in the acute phase of ischemic stroke consists of intravenous thrombolysis (IVT, through recombinant tissue- type plasminogen reactivator) and/or endovascular treatment (EVT). The challenging part is that IVT is provided in almost all hospitals (primary stroke centres), but that EVT is a highly specialized treatment only provided in a few hospitals (comprehensive stroke centres). All of these parties need to share information in the acute setting while treating the patient. Furthermore, earlier research has shown that the earlier the treatment has been given the better functional outcomes are for the patient [5]. Therefore, a break-glass access mechanism to improve data availability has potential benefits.

When a patient suffers a stroke, the patient itself, a family member or the general practitioner is the first to contact the emergency call centre. During the telephone call, trained healthcare workers follow a triage system where a suspected stroke may be concluded. When an ambulance is sent to the patient, the goal is to arrive within 45 min. When an ambulance goes to the patient, information already collected by the emergency call centre is sent by messages and displayed in a fixed device inside the ambulance (e.g. age, gender). Once the ambulance arrives, the ambulance team examines the patient and collects more data (e.g. blood pressure, pulse, oxygen saturation, glucose). When the ambulance team suspects a stroke and decides to take the patient to the closest hospital, it contacts this hospital by phone to inform the estimated arrival time. When the ambulance team arrives at the hospital, all information they collected will be presented to the hospital team orally. After delivering the patient, they fill the collected data into an electronic form on their tablet for recording purposes, but this will be too late to turn available for the hospital team.

After the phone call from the ambulance nurse, the concerned hospital get prepared for the patient. The neurologist or resident on call, the neurology nurse, the emergency doctor and nurse, the radiologist and the radiology technician will clear the room for image exams and wait for the patient. If the patient already has a medical record in the hospital, it is evaluated. If not, a new patient identification number will be created to store the new data. Furthermore, the doctor will try to call other hospitals or the patient's general practitioner to obtain more information about the patient. If a patient is eligible to EVT, and she needs to be transferred to a comprehensives stroke centre, all collected information is shared both orally and by e-mail between the sending and receiving hospitals. In this case, the patient is transferred by a second ambulance, which also needs the available information. At last, when the patient receives the EVT, a team of medical doctors await, including the neuro-interventional radiologist, radiology technician, and anesthesiologist, that also need to know all information. After transportation, all collected data is presented to the doctors one more time, orally. Imaging data have been sent through an imaging-exchange system for the neuro-interventionist and radiologist.

Note that three or more teams are involved in the treatment, requiring access to the patient's EMR and generating new content for it. Between all those moments of consultation, data can be missed or forgotten to mention. Therefore, to improve accessibility to medical records and protect patient's privacy, it is necessary to dynamically grant and revoke access to the patient's EMR.

4 Cryptographic primitives

Here we define the basic cryptographic primitives used throughout the paper and define a CP-ABE scheme as following [17].

The set of all binary strings of length n is denoted by $\{0,1\}^n$, and the set of all finite binary strings as $\{0,1\}^*$.

Given a set V, we refer to the ith element as v_i. Additionally, we use the following notations for cryptographic operations throughout the paper:

- For an arbitrary message $m \in \{0,1\}^*$, $c = \mathrm{Enc}(K, m)$ denotes a symmetric encryption of m using the secret key $K \in \{0,1\}^*$, and $m = \mathrm{Dec}(K, c) = \mathrm{Dec}(K, \mathrm{Enc}(K, m))$ is the corresponding symmetric decryption operation.
- We denote by pk/sk a public/private key pair for an IND-CCA2 secure public key encryption scheme PKE. An encryption of message m under the public key pk is denoted by $c = \mathrm{Enc}_{pk}(m)$ and the corresponding decryption operation by $m = \mathrm{Dec}_{sk}(c) = \mathrm{Dec}_{sk}(\mathrm{Enc}_{pk}(m))$.
- $\sigma = \mathrm{Sign}_{sk}(m)$ denotes a EUF-CMA secure digital signature over a message m. The corresponding verification operation for a digital signature is denoted by $b = \mathrm{Verify}_{pk}(m, \sigma)$, where $b = 1$ if the signature is valid, and $b = 0$ otherwise.
- A one-way hash function (H) over a message m is denoted by $H_m = H(m)$.
- We denote by $r = \mathrm{RAND}(n)$ a random binary sequence of length n, where RAND(n) represents a random function that takes a binary length argument n as input and gives a random binary sequence of this length in return.[1]

A CP-ABE scheme is a tuple of the following four algorithms:

1. CPABE.Setup is a probabilistic algorithm that takes as input a security parameter λ and outputs a master public key MPK and a master secret key MSK. We denote this by $(\mathrm{MPK}, \mathrm{MSK}) \leftarrow \mathrm{Setup}(1^{\lambda})$.
2. CPABE:Gen is a probabilistic algorithm that takes as input a master secret key, a set of attributes $A \in \Omega$ and the unique identifier of a user, and it outputs a secret key that is bound both to the corresponding list of attributes and the user. We denote this by $(\mathrm{sk}_{A,i}) \leftarrow \mathrm{Gen}(\mathrm{MSK}, A, u_i)$.
3. CPABE:Enc is a probabilistic algorithm that takes as input a master public key, a message m and a policy $P \in P$. After a proper run, the algorithm outputs a ciphertext c_P which is associated to the policy P. We denote this by $c_P \leftarrow \mathrm{Enc}(\mathrm{MPK}, m\ P)$.
4. CPABE:Dec is a deterministic algorithm that takes as input a user's secret key and a ciphertext and outputs the original message m *iff* the set of attributes A that are associated with the underlying secret key satisfies the policy P that is associated with c_P. We denote this by $\mathrm{Dec}(\mathrm{sk}_{A,i}, c_P) \rightarrow m$.

[1] We assume that a true random function is replaced by a pseudo-random function, the input-output behaviour of which being 'computationally indistinguishable' from that of a true random function.

5 System model

The system model presented here is based on the model introduced in [18]. Below we present an overview of the main entities of the system and the most relevant communication between them.

Cloud service provider (CSP) The cloud computing environment is based on a trusted Infrastructure-as-a-Service (IaaS) provider. The IaaS platform consists of cloud hosts that operate virtual machine (VM) guests and communicate through a network. In our model, we require that the IaaS runs a protocol similar to the one described in [19], where the integrity of the underlying CSP is verified. In principle, such integrity verification can be added to any IaaS. A CSP stores patients' EMR encrypted under a CP-ABE scheme. Additionally, the CSP is responsible for controlling the access to the encrypted EMR.

Registration authority (RA) The RA is responsible for the registration of all healthcare entities and users. The RA generates user attributes that will be used for the proper authorization (e.g. membership to a particular treatment team). The RA can run as a separate third party, but can be also implemented as part of the CSP. The registration process is out the scope of this work.

Master authority (MA) The MA has a master secret key MSK and a public key MPK. The master key is kept private, while the public key is known to everyone. Additionally, the MA uses MSK to generate CP-ABE secret keys for users based on her attributes to authorize access to an encrypted EMR. The MA is also responsible for granting and revoking tokens used for dynamic access control.

User We consider three different types of users: patients, healthcare professionals and healthcare entities. The set of all patients registered at RA is denoted by $U = \{u_1, \ldots, u_{N_u}\}$ and the set of all registered healthcare professionals is denoted as $S = \{s_1, \ldots, s_{N_s}\}$. A healthcare entity is a special type of user represented by an attested smart device. This device serves to confirm the following treatment team locations: Emergency Call Centre (e), Ambulance (a) and Hospital (h). A treatment team is a group of professionals co-located at one of the entities that attest each other's involvement in the emergency situation. Each user from U, S and the healthcare entities has a unique public/private key pair (pk/sk) used to communicate securely through an IND-CCA2 secure public key encryption scheme PKE and an EUF-CMA secure signature scheme sign.

6 Problem statement and threat model

6.1 Problem statement

Let u_i be a patient from the set U and $s_j \in S$ be a member of one of the stroke treatment teams. Let us assume that u_i has a set of N different files stored in the CSP. We denote this set of files as $D_i = \{d_1^i, \ldots, d_N^i\}$. The problem is to find a way to achieve the following:

1. Enable access to the content of each $d_l^i \in D_i$ to s_j involved in the treatment of u_i;
2. User s_j has access to D_i if and only if she has a legitimate role in the treatment team of u_i at the time, as given by a valid policy;
3. Access control to D_i should be granted and revoked dynamically as requested for the patient's treatment. This should not require to decrypt and re-encrypt the file with a fresh key, and it should not affect the access by the rest of the legitimate users.

6.2 Threat model

Our threat model is similar to the one described in [19], which is based on the Dolev-Yao adversarial model [20]. We further assume that privileged access rights can be used by a remote adversary *ADV* to leak confidential information. *ADV*, e.g. a corrupted system administrator, can obtain remote access to any host maintained by the CSP, but cannot access the volatile memory of guest VMs residing on the compute hosts of the CSP. Moreover, we extend the above threat model by defining a set of attacks available to *ADV*.

Attack 1 (Token Alteration Attack). *Let ADV be a corrupted user that has been legitimately part of a treatment team in the past. ADV successfully launches a Token Alteration Attack if she can modify a token she received in the past in such a way that it will be considered as valid by the CSP.*

Attack 2 (Token Substitution Attack). *Let ADV be a corrupted user who overhears all communication and captures a token issued for another legitimate healthcare professional. ADV successfully launches a Token Substitution Attack if she can use this token to add false ciphertexts to the patient's EMR.*

Attack 3 (Revocation of Legitimate Users Attack). *Let ADV be a corrupted user. Moreover, let x be a team who currently has access to patient's ui EMR. ADV successfully launches a Revocation of Legitimate Users Attack if she can manage to revoke team x from accessing the patients encrypted data.*

7 Red Alert Protocol

We propose 'Red Alert Protocol' (RAP) for the problem presented in Section 6. More precisely, our approach follows the protocols proposed in [18, 21], with additions to meet the specific needs of the acute stroke care case described in Section 3. RAP was initially presented in [22], but here we extend that work by revising the protocol to address a broader threat model and presenting the protocol in a more formal construction.

Below we first present an overview of the protocol followed by its definition.

7.1 Protocol overview

We assume that each user (from U or S) is registered through a central RA. However, we consider registration as out of the scope of this paper and assume that all users have been previously registered. Each user receives a unique identifier i, and a set of attributes A is created based on the user's personal data. For patients, identifying attributes such as name and surname could be used. For healthcare professionals, attributes include identification and function in the organization, and in particular the membership and role in an emergency treatment team. Also, we assume that the EMRs are in a standardized and interoperable format before being encrypted and stored in the CSP.

RAP is divided into *Setup* and four main phases: *Initialization, Emergency Session, Process Data* and *Leave Session*. Figure 1 shows the messages exchanged between the entities in each phase.

During the *Initialization* phase, a patient u_i stores her EMR on the CSP as a ciphertext c_P^i. In this paper, we explicitly focus on the problem of how only authorized users can access a patient's EMR during an emergency session. To this end, the policy P needs to always contain a condition that will allow a user s_j to successfully decrypt $d_l^i \in D_i, \forall \int \in [1, |D_i|]$. Among other conditions in P, the following should be added for u_i: '… OR (Emergency = TRUE AND TreatmentTeamMember = TRUE AND UserInEmergency = i)'. A professional s_j will be then granted access to the EMR of u_i only when her attributes satisfy this policy.

In the *Emergency Session* phase, the MA associates the patient to all the treatment teams involved in her emergency session, which ends after complete treatment and patient discharge. The session starts when a patient, or someone on her behalf, contacts the call centre team by phone. Figure 2 shows the patient timeline during the emergency care. The call centre professional $s_e \in S$ requests MA to initiate the emergency session; s_e also

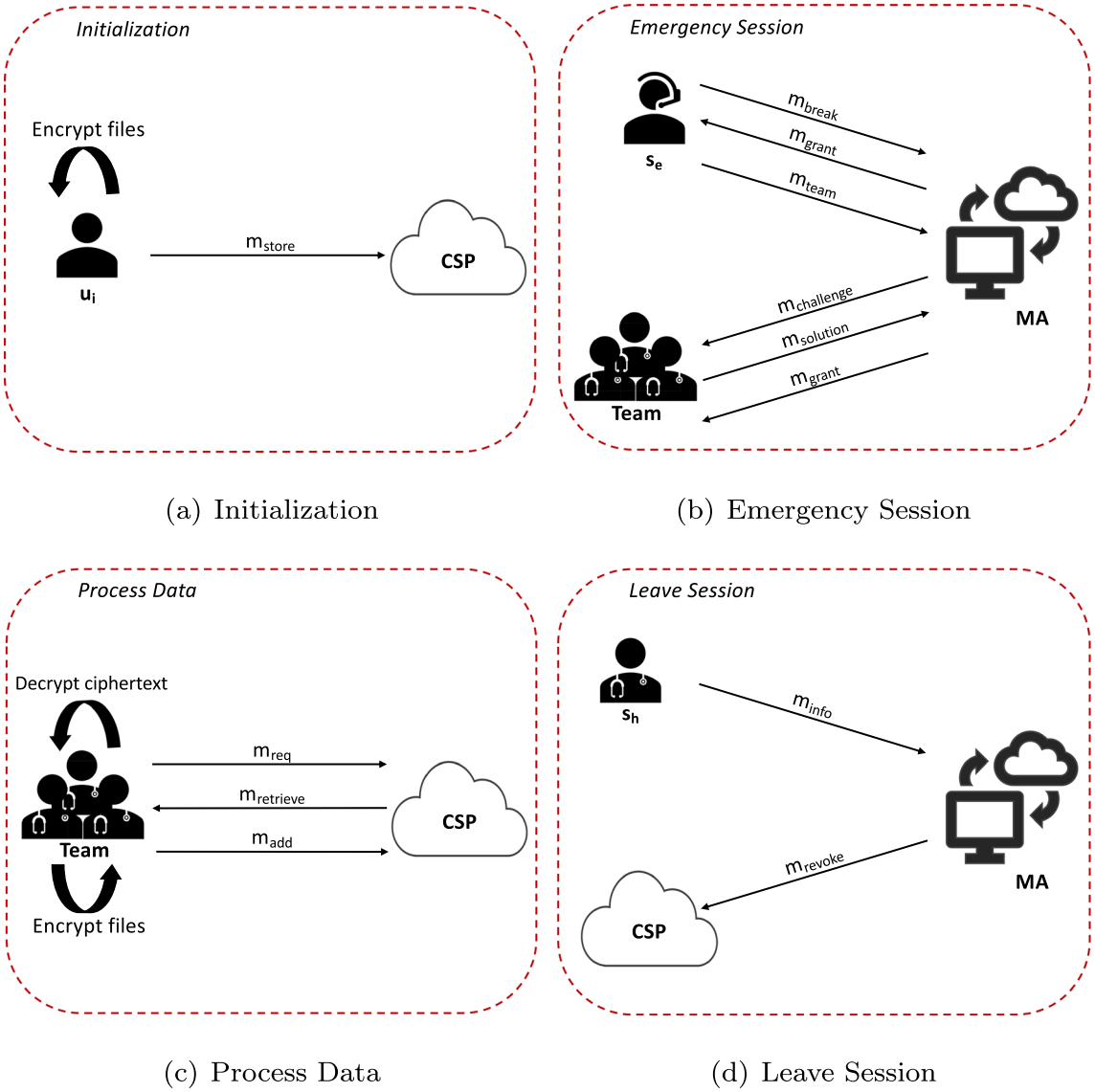


Fig. 1 Overview of the Red Alert Protocol: entities and their communication during the four phases

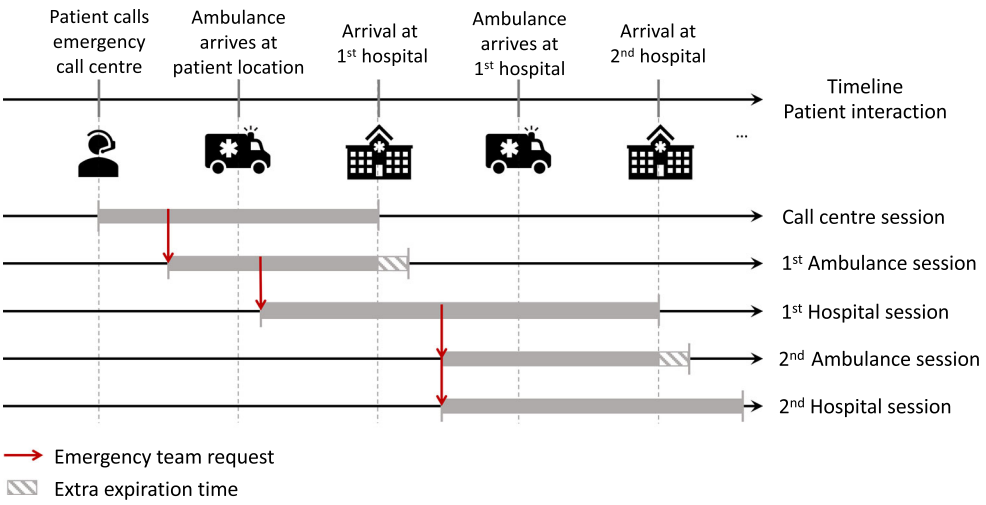


Fig. 2 Stroke care and data access session timelines. The top line represents events of interaction between healthcare provider entities and the patient. The others show the period of time that each team has access to the patient data

involves the ambulance team in the session, which ultimately will also involve the hospital treatment team. In this proposal we trust that s_e will contact the MA only if she receives a legitimate phone call from the patient or someone on behalf of the patient. The phone call authentication is very important, but it is considered outside the scope of this paper. However, each involved treatment team needs to prove to the MA that their service is requested. This is done when the treatment team jointly solves a challenge: the healthcare entity x and at least two[2] users respond to the challenge, proving that they are co-located and working together. After the challenge is solved and the users' attributes are validated, the MA generates a CPABE emergency key. As attributes, among others, MA inserts in A_e the following: '[Emergency, TreatmentTeamMember, i]'. This guarantees that the generated key will satisfy the policy bound to u_i's ciphertexts. However, direct sharing the CPABE emergency key is not secure enough, because getting access to that key would allow anyone to access $u_i's$ ciphertexts at any future moment. Therefore, the MA also generates an access control token τ_x to the team. This token has a default expiration time and also contains the identity of the professionals from the treatment team. The MA subsequently sends the key and token to the team.

In the *Process Data* phase, one of the professionals in the team sends the access token to the CSP to retrieve the patient data. If the token is valid, the CSP grants access to retrieve the ciphertext containing the EMR of the patient under emergency treatment. Through a secure read-only application, the EMR is decrypted by the professional using the CPABE emergency key. Token validation also takes place when s_j adds new data to the patient's EMR by uploading a new ciphertext to the CSP.

The *Leave Session* phase takes place as soon as the patient is no longer under care of a treatment team. To do so, the MA needs to be informed about the current location of the patient by either a check-in or a check-out message. Both messages are sent by the attested smart devices of each treatment team and include a timestamp. These messages can be implemented according to the application. With this information, the MA can revoke the token of the previous team that is no longer involved in the treatment. With this information, the MA can revoke the token of the previous team that is no longer involved in the treatment. In stroke acute care, the moments when the patient arrives and leaves the hospital emergency care unit define the end of involvement of treatment teams. When the patient arrives at the first hospital, the call centre and ambulance teams leave the emergency session. For the call centre, revocation of τ_e should be immediate. The ambulance team, however, is granted extra time after arrival at the hospital to add their reports into the medical record (see Fig. 2). In principle, τ_h needs to be revoked when the patient leaves the hospital emergency care. However, if the patient needs to be transferred for treatment, the token for the first hospital will be revoked as soon as the patient arrives at the second hospital. As soon as the MA knows the moment when the patient arrived or left the hospital emergency care, it sends a revocation message for the corresponding access token to the CSP. Thus, even if a token is still valid according to the default expiration time, the CSP will not allow any type of access to the data after the revocation time.

The emergency session ends when all tokens associated with it have expired or explicitly revoked. After this, no new team is allowed to join the session anymore.

7.2 Protocol definition

The 'Red Alert Protocol' (RAP) defines the exchange of messages to grant and revoke access, as well as to rightfully encrypt and decrypt the patient's EMR during an emergency session. In all cases, the entity receiving the message verifies the freshness and the integrity of the message, and it can also authenticate the sender through a signature.

During the phases, all the entities and users interact by running the following algorithms: RAP.Setup, RAP.StoreData, RAP.GrantAccess, RAP.BreakGlass, RAP.JoinTeam, RAP.RetrieveData, RAP.AddData and RAP.RevokeAccess. The phases and algorithms are detailed below as follows: the algorithms are described inside frames in each phase where they are used. An

Table 1 Protocol messages

Index	Message
m_{store}	$\langle r_1, c_P^i, \sigma_i(H(r_1 \| c_P^i)) \rangle$
m_{grant}	$\left\langle r_3, \tau_x, \mathrm{Enc}_{\mathrm{pk}_{s_x}}(\mathrm{sk}_{A_e,\mathrm{i}}), \sigma_{\mathrm{MA}}(H(r_3 \| \tau_x \| \mathrm{sk}_{A_e,\mathrm{i}})) \right\rangle$
m_{break}	$\langle r_4, \mathrm{Enc}_{\mathrm{pk}_{\mathrm{MA}}}(u_i, t, s_e), \sigma_{s_e}(H(r_4 \| u_i \| t \| s_e)) \rangle$
m_{team}	$\langle r_5, \mathrm{Enc}_{\mathrm{pk}_{\mathrm{MA}}}(x, s_{x_1}, s_{x_2}), \sigma_{s_j}(H(r_5 \| x \| s_{x1} \| s_{x2})) \rangle$
$m_{challenge}$	$\langle r_6, \mathrm{challenge}, \sigma_{\mathrm{MA}}(H(r_6 \| \mathrm{challenge})) \rangle$
$m_{solution}$	$\langle r_7, \mathrm{Enc}_{\mathrm{pk}_{\mathrm{MA}}}(v_0, l_x), \sigma_{\mathrm{MA}}(H(r_7 \| v_0 \| s_{x1} \| s_{x2} \| l_x)) \rangle$
m_{req}	$\langle r_8, \tau_x, \sigma_{s_j}(H(r_8 \| \tau_x)) \rangle$
$m_{retrieve}$	$\langle r_9, c_P^i, \sigma_{\mathrm{CSP}}(H(r_9 \| c_P^i)) \rangle$
m_{add}	$\langle r_{10}, \tau_x, c_P^i, \sigma_{s_j}(H(r_{10} \| \tau_x \| c_P^i)) \rangle$
m_{info}	$\langle r_{11}, \mathrm{E}_{\mathrm{pk}_{\mathrm{MA}}}(t), \sigma_{s_h}(H(r_{11} \| t)) \rangle$
m_{revoke}	$\langle r_{12}, \tau_x, \sigma_{\mathrm{MA}}(H(r_{12} \| \tau_x)) \rangle$

[2] Here we assume that at least two professionals are part of the team, but more could be included in similar way.

overview of all messages exchanged during the execution of the algorithms is presented in Table 1.

Setup Before start, each system model entity denoted as ID (for MA, RA, CSP, and users) runs the algorithm RAP.Setup. The entities obtain a public/private key pair (pk, sk) for a IND-CCA2 secure public cryptosystem PKE and publish their public key while keeping their private key secret. Furthermore, MA runs CPABE.Setup to acquire a master public/private key pair (MPK, MSK) and publishes the public master key.

RAP.Setup

Input $(\text{"}initialize\text{"}, 1^{\lambda})$.

Output $(\mathsf{pk_{ID}}, \mathsf{sk_{ID}})$, $(\mathsf{MPK}, \mathsf{MSK})$

1. ID runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(1^{\lambda})$.
2. ID publishes $\mathsf{pk_{ID}}$.
3. MA runs $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{CPABE.Setup}(1^{\lambda})$.
4. MA publishes MPK.

Initialization phase In this phase, the patient runs RAP.StoreData to encrypt her EMR using CPABE and an emergency policy. After u_i successfully encrypts her data, she sends her c_P^i to the CSP.

RAP.StoreData

Input A collection of files d_l^i, MPK, emergency policies P.

Output A collection of ciphertexts c_P^i stored on the CSP.

1. u_i runs $c_P^i \leftarrow \mathsf{CPABE.Enc}(\mathsf{MPK}, d_l^i, P)$
2. u_i generates a random number r_1
3. u_i sends $m_{store} = \langle r_1, c_P^i, \sigma_i(H(r_1 || c_P^i)) \rangle$ to CSP
4. CSP verifies m_{store}, if the verification fails, output $\perp$.
5. CSP stores c_P^i

Emergency session In this phase health professionals involved in an emergency session obtain access to patient data through three algorithms: RAP.BreakGlass, RAP.JoinTeam and RAP.GrantAccess.

RAP:GrantAccess: The MA multiple times generates an access token and a CPABE emergency key for each treatment team s_x, where $x \in \{e, a, h\}$.

RAP.GrantAccess

Input u_i and the team involved in the emergency session (s_x); MSK and emergency attributes $\mathcal{A}_e$

Output Access Token (τ_x) and CPABE emergency key ($\mathsf{sk}_{\mathcal{A}_e,\mathsf{i}}$)

1. MA calculates the default expiration time: t_{exp}
2. MA generates the timestamp: t_{gen}
3. MA generates a random number r_2
4. MA generates $\tau_x = (t_{gen}, t_{exp}, \mathsf{Enc}_{\mathsf{pk}_{\mathrm{CSP}}}(r_2, s_{x1}, s_{x2}, u_i), \sigma_{\mathrm{MA}}(H_\tau))$
5. MA runs $\mathsf{sk}_{\mathcal{A}_e,\mathsf{i}} \leftarrow \mathsf{CPABE.Gen}(\mathsf{MSK}, \mathcal{A}_e, u_i)$
6. MA generates a random number r_3
7. MA sends $m_{grant} = \langle r_3, \tau_x, \mathsf{Enc}_{\mathsf{pk}_{s_x}}(\mathsf{sk}_{\mathcal{A}_e,\mathsf{i}}), \sigma_{\mathrm{MA}}(H(r_3||\tau_x||\mathsf{sk}_{\mathcal{A}_e,\mathsf{i}}))\rangle$ to the team s_x
8. s_x verifies m_{grant}, if the verification fails, output $\perp$.
9. s_x recovers τ_x and $\mathsf{sk}_{\mathcal{A}_e,\mathsf{i}}$

RAP:BreakGlass: Through this process the MA acknowledges the emergency event for a patient u_i and begins the emergency session. After identifying patient u_i, s_e contacts MA to notify the emergency event and requests to become part of the emergency session. Upon reception, MA confirms that s_e is indeed part of the call centre team and runs RAP.GrantAccess.

RAP.BreakGlass

Input u_i, emergency call to s_e

Output MA runs $\mathsf{RAP.GrantAccess}(u_i, s_e, \mathsf{MSK}, \mathcal{A})$

1. s_e records the time of the call: t
2. s_e generates a random number r_4
3. s_e sends $m_{break}\langle r_4, \mathsf{Enc}_{\mathsf{pk}_{\mathrm{MA}}}(u_i, t, s_e), \sigma_{s_e}(H(r_4||u_i||t||s_e)\rangle$ to MA
4. MA verifies m_{break}, if the verification fails, output $\perp$.
5. MA checks if $s_e \in \mathcal{S}$; if not, output $\perp$.
6. MA start an emergency session for u_i
7. MA includes s_e in the emergency session

RAP.JoinTeam: The MA associates users in a treatment team to an existing emergency session. After the team authentication by solving the challenge, MA includes all the team members x, s_{x1}, s_{x2} into the emergency session and runs RAP.GrantAccess.

RAP.JoinTeam

Input The identities of the new team members x, s_{x1}, s_{x2}, where $x \in \{a, h\}$, and u_i

Output MA runs $\mathsf{RAP.GrantAccess}(u_i, s_{x1}, s_{x2}, \mathsf{MSK}, \mathcal{A}_e)$

1. s_j generates a random number r_5
2. s_j sends $m_{team} = \langle r_5, \mathsf{Enc}_{\mathsf{pk}_{\mathrm{MA}}}(x, s_{x_1}, s_{x_2}), \sigma_{s_j}(H(r_5||x||s_{x1}||s_{x2}))\rangle$ to MA
3. MA check if $s_j \in \mathcal{S}$
4. MA verifies m_{team}; if the verification fails, output $\perp$.
5. MA checks if $(x, s_{x1}, s_{x2}) \in \mathcal{S}$; if not, output $\perp$.
6. MA generates a random number v
7. MA splits in three random shares: $v = v_0 + v_1 + v_2$
8. MA generates $\mathsf{challenge} = \langle \mathsf{Enc}_{\mathsf{pk}_x}(v_0), \mathsf{Enc}_{\mathsf{pk}_{s_{x1}}}(v_1), \mathsf{Enc}_{\mathsf{pk}_{s_{x2}}}(v_2)\rangle$
9. MA generates a random number r_6
10. MA sends $m_{challenge} = \langle r_6, \mathsf{challenge}, \sigma_{\mathrm{MA}}(H(\mathsf{challenge}||r_6))\rangle$ to x, s_{x1} and s_{x2}
11. x, s_{x1} and s_{x2} recover their own share, respectively v_0', v_1' and v_2'
12. x, s_{x1} and s_{x2} record their own location, respectively l_x, l_{x1} and l_{x2}
13. x generates a random number r_7
14. x sends $m_{solution} = \langle r_7, \mathsf{Enc}_{\mathsf{pk}_{\mathrm{MA}}}(v_0', l_x), \sigma_x(H(r_7||v_0'||s_{x1}||s_{x2}||l_x))\rangle$ to MA
15. s_{x1} and s_{x2} send analogous $m_{solution}$ messages to MA
16. MA verifies all $m_{solution}$; if any verification fails, output $\perp$.
17. MA calculates $v' = v_0' + v_1' + v_2'$
18. MA verifies if $v = v'$, if not output $\perp$.
19. MA verifies if $l_x = l_{x1} = l_{x2}$, if not output $\perp$.
20. MA includes x, s_{x1}, s_{x2} in the emergency session

Process data phase After having received the CPABE emergency key and an access token, s_j from team $x \in \{e, a, h\}$ is ready to process the patient's data through either RAP.RetrieveData or RAP.AddData.

RAP.RetrieveData: First s_j requests the CSP to retrieve all ciphertexts in the EMR for the patient under emergency treatment. After successful message and token verification, the CSP sends the elegible u_i's ciphertexts to s_j. Finally, s_j uses the CPABE emergency key $\mathrm{sk}_{Ae,\ i}$ to recover the data from c_P^i.

RAP.RetrieveData

Input τ_x

Output A collection of files d_l^i from patient's EMR

1. s_j generates a random number r_8
2. s_j sends $m_{req} = \langle r_8, \tau_x, \sigma_{s_j}(H(r_8||\tau_x))\rangle$ to MA
3. CSP verifies m_{req}, if the verification fails, output $\perp$.
4. CSP verifies if τ_x is valid, if not output $\perp$.
5. CSP generates a random number r_9.
6. CSP sends $m_{retrieve} = \langle r_9, c_P^i, \sigma_{\mathrm{CSP}}(H(r_9||c_P^i))\rangle$ to s_j
7. s_j runs $d_l^i \leftarrow \mathsf{CPABE.Dec}(\mathsf{sk}_{\mathcal{A}_e,i}, c_P^i)$

RAP.AddData: During and after patient's treatment, all teams may upload new files d_l^i to the patient EMR. The same policy P needs to be used as in the already existing encrypted EMR.

RAP.AddData

Input MPK, d_l^i, P and τ_x

Output A new ciphertext c_P^i stored on the CSP

1. s_j runs $c_P^i \leftarrow \mathsf{CPABE.Enc}(\mathsf{MPK}, d_l^i, P)$
2. s_j generates a random number r_{10}.
3. s_j sends $m_{add} = \langle r_{10}, \tau_x, c_P^i, \sigma_{s_j}(H(r_{10}||\tau_x||c_P^i))\rangle$ to CSP
4. CSP verifies m_{store}, if the verification fails, output $\perp$.
5. CSP verifies if τ_x is valid, if not output $\perp$.
6. CSP stores c_P^i.

Leave session As soon as the patient arrives or leaves the hospital, s_h initiates RAP.RevokeAccess. Subsequently, MA calculates the time to revoke the tokens from the teams which are no longer needed for treatment (see Fig. 2), the MA sends the respective τ_x to be revoked to CSP.

RAP.RevokeAccess

Input Patient current location

Output τ_x revoked

1. s_h records the time of arriving/leaving the hospital: t
2. s_h generates a random number r_{11}
3. s_h sends $m_{info} = \langle r_{11}, \mathsf{E}_{\mathsf{pk}_{\mathrm{MA}}}(t), \sigma_{s_h}(H(r_{11}||t)) \rangle$ to MA
4. MA verifies m_{info}, if the verification fails, output $\perp$
5. MA checks if $s_h \in \mathcal{S}$, if not output $\perp$
6. MA calculates the time to revoke the tokens
7. MA generates a random number r_{12}
8. MA sends $m_{revoke} = \langle r_{12}, \tau_x, \sigma_{\mathrm{MA}}(H(r_{12}||\tau_x)) \rangle$ to CSP
9. CSP verifies m_{revoke}, if the verification fails, output $\perp$.
10. CSP revokes τ_x

8 Security analysis

8.1 Simulation-based security

To prove the security of our protocol, we assume the existence of a simulator coined *SIM*. It will simulate the algorithms from the real protocol in a way that any polynomial time adversary *ADV* will not be able to distinguish between the real algorithms and the simulated ones.

Definition 1 (Sim-Security). *We consider the following experiments. In the real experiment, all algorithms run as specified in our protocol. In the ideal experiment, SIM intercepts ADV's queries and replies with simulated responses.*

Real Experiment	*Ideal Experiment*
1. $\mathbf{EXP}^{real}_{\mathsf{RAP}}(1^\lambda)$	1. $\mathbf{EXP}^{ideal}_{RAP}(1^\lambda)$
2. $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{RAP.Setup}(1^\lambda)$	2. $(\mathsf{MPK}) \leftarrow \mathcal{SIM}(1^\lambda)$
3. $\mathsf{sk}_{\mathcal{A}_e,\mathsf{u_i}} \leftarrow \mathcal{ADV}^{\mathsf{RAP.GrantAccess}(\mathsf{MSK},\mathcal{A}_e)}$	3. $\mathsf{sk}_{\mathcal{A}_e,\mathsf{u_i}} \leftarrow \mathcal{ADV}^{\mathcal{SIM}(1^\lambda)}$
4. $ct \leftarrow \mathsf{CPABE.Enc}(\mathsf{MPK}, d^i_l)$	4. $ct \leftarrow \mathcal{SIM}(1^\lambda, 1^{\lvert d^i_l \rvert})$
5. *Output* b	5. *Output* b'

We say that RAP *is sim-secure if for all Probabilistic Polynomial Time (PPT) adversaries ADV:*

$$\mathbf{EXP}^{\,real}_{\mathrm{RAP}}(1^\lambda) \approx \mathbf{EXP}^{\,ideal}_{\mathrm{RAP}}(1^\lambda)$$

Theorem 1. *Assuming that PKE is an IND-CCA2 secure public key cryptosystem and* Sign *is an EUF-CMA secure signature scheme, then* RAP *is a sim-secure protocol according to Definition 1.*

Proof. We start by defining the algorithms used by the simulator (identified with *). Then, we will replace the real algorithms with the simulated ones. Finally, with the help of a Hybrid Argument, we will prove that the resulted distributions are indistinguishable.

1. RAP.Setup*: Will only generate MPK that will be given to ADV.
2. RAP.StoreData*: Will simulate a ciphertext that has the same length as the output of the real algorithm.

3. RAP.GrantAccess*: Will generate a random string to be sent to *ADV*. The random string has the same length as the output of the real algorithm. Moreover, *SIM* simulates a token τ^* that has the same length as the real token τ.
4. RAP.RetrieveData*: Will return the specified file, without running the decryption protocol.

The RAP.BreakGlass* and RAP.JoinTeam* oracles are included in the RAP.GrantAccess* one. The reason for this is that, during the execution of both of these algorithms, *ADV* queries RAP.GenKey and RAP.GenToken. Moreover, since RAP.Revoke does not produce any output, we can exclude it from our proof. Finally, RAP.AddData can be seen as a special case of RAP.StoreData. As a result, by proving that StoreData is secure, we also prove that RAP.AddData is secure. In a preprocessing phase, *SIM* creates a list L in which it will store the files used by *ADV* in RAP.StoreData.

Hybrid 0 RAP runs normally.

Hybrid 1 Everything runs like Hybrid 0, but we replace RAP.Setup with RAP.Setup*.

These algorithms are identical from the *ADV*'s perspective, and as a result, the Hybrids are indistinguishable.

Hybrid 2 Everything runs like Hybrid 1, but we replace RAP.StoreData with RAP.StoreData*.

At this point, *SIM* will simulate a ciphertext that will be sent to *ADV*. Moreover, *SIM* will store in L the tuple $(d_l^i, c_p^{i\,*})$ where d_l^i is a file that was given as input by *ADV* and $c_p^{i\,*}$ is the simulated ciphertext that corresponds to d_l^i. The Hybrids are indistinguishable from *ADV*'s point of view, since she receives what she believes to be a valid ciphertext.

Hybrid 3 Everything runs like Hybrid 1, but we replace RAP.BreakGlass with RAP.BreakGlass*.

Again the algorithms are identical from *ADV*'s point of view and thus, the Hybrids are indistinguishable.

Hybrid 4 Everything runs like Hybrid 2, but we replace RAP.RetrieveData with RAP.RetrieveData*.

At this point, *SIM* retrieves L, finds the d_l^i that corresponds to the c_p^i that was given as input by *ADV*, and returns it. Clearly, since *ADV* receives the file she was waiting for, the Hybrids are indistinguishable.

With this Hybrid, our proof is complete. We managed to replace the expected outputs with simulated responses in a way that *ADV* cannot distinguish between the real and the ideal experiment.

8.2 Protocol security

In this section, we prove the resilience of our protocol against the set of attacks defined in Section 6.2. We assume that the random numbers r_i generated throughout the protocol are stored locally on each entity. To ensure that r_i is used only once, it is time-variant, including a suitably fine-grained timestamp in its value. In this way, we can guarantee the freshness of the exchanged messages.

Proposition 1 (Token Alteration Attack Soundness). *Let ADV be a corrupted user and u_i be a patient whose EMR is stored in the CSP. Moreover, we assume that ADV's access to u_i's EMR has been revoked. Then, ADV cannot successfully perform a Token Alteration Attack.*

Proof. Since *ADV*'s access is revoked, it is implied that at some point in the past *ADV* received a valid token $\tau_x = \left(t_{gen}, t_{exp}, Enc_{\mathrm{pk_{CSP}}}(r, s_{x_1}, s_{x_2}, u_i), \sigma_{MA}(H(\tau_x))\right.$ to access the medical records of a user u_i. As a result, all *ADV* needs to do in order to launch a Token Alteration Attack is to modify the timestamps contained in τ_x. However, since the timestamps are also contained in the hash of the signature of MA, altering the timestamps is equivalent to forging MA's signature, which, given the EUF-CMA security of the signature scheme, can only happen with negligible probability. Therefore, the attack fails.

Proposition 2 (Token Substitution Attack Soundness). *Let ADV be a corrupted user who overhears all communication and captures a token τ issued to another legitimate healthcare professional s_l. Then ADV cannot successfully launch a Token Substitution Attack.*

Proof. *ADV* can capture the token τ by overhearing the messages m_{grant}, m_{req} and m_{add}. However, the intercepted τ contains the identity of s_l, which cannot be changed because *ADV* cannot generate a valid signature as we already proved; therefore, the protocol is secure against a Token Alteration Attack. The only alternative for *ADV* is to use τ directly. To this end, *ADV* runs $c_P^i \leftarrow \mathrm{CPABE.Enc}\left(\mathrm{MPK}, d_l^i, P\right)$ for a fake d_l^i, in an attempt to create a valid m_{add} message. However, for *ADV* to successfully create a m_{add}, she also needs to forge s_l's signature, which can only happen with negligible probability since we assume that the signature scheme is EUF-CMA secure. As a result, the attack will fail.

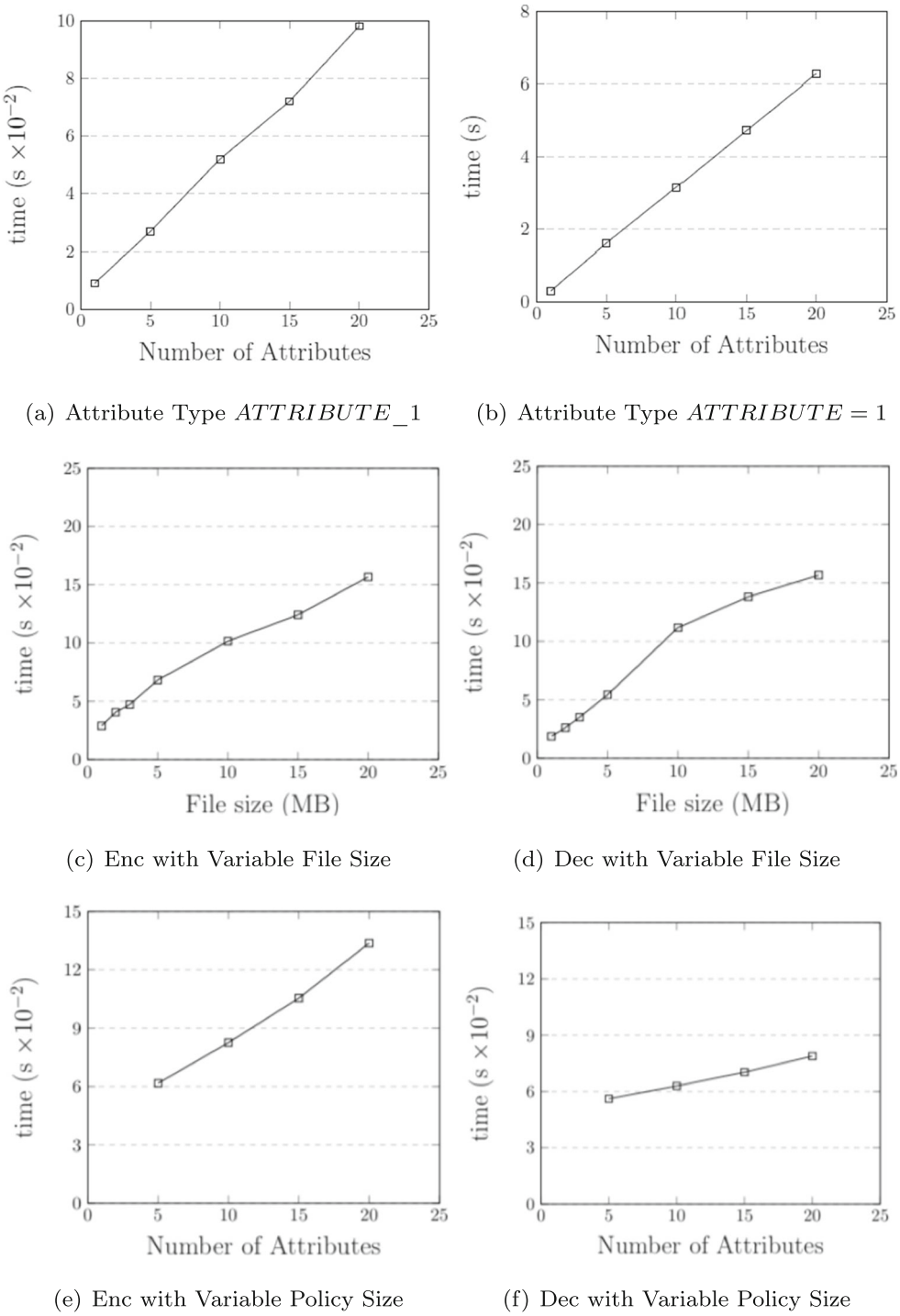
Proposition 3 *Let ADV be a corrupted user. ADV cannot successfully launch a Revocation of Legitimate Users Attack.*

Proof. *ADV* commences the attack by trying to construct a valid $m_{info} = \left\langle r_{11}, \mathrm{E}_{\mathrm{pk_{MA}}}(t), \sigma_{sh} H((r_{11} \| t)) \right\rangle$ message for MA. However, this message needs to be signed by a s_h who is already a legitimate member of the team. As a result, creating a valid m_{info} message is equivalent to forging s_h's signature, which can only happen with negligible probability, since the signature scheme is EUF-CMA secure. The only other option for *ADV* is to bypass MA and try to communicate directly with the CSP. To this end, *ADV* tries to construct a valid $m_{revoke} = \langle r_{12}, \tau_x, \sigma_{\mathrm{MA}}(H(r_{12} \parallel t_x)) \rangle$. However, once again, the EUF-CMA security will prevent *ADV* from forging MA's signature, and as a result, the attack will fail.

9 Experimental results

In this section, we present based on the implementation of the core functions of our protocol. We prove the effectiveness of the proposed protocol by evaluating the processing time of the core functions on a standalone Linux machine. Our experiments mainly focused on the key generation phases in RAP.Setup and RAP.GrantAccess, encryption, decryption, signing, and verification functions. For the encryption/decryption and signing/verifying we used the RSA cryptosystem, and for the attribute-based encryption scheme we used the CPABE library provided in [23]. Finally, SHA256 used as the main cryptographic hash function.

The experiments were carried out on an Intel Core i7-4790 CPU @ 3.60 GHz x8 Ubuntu 18.04.2 Desktop with 16 GB of RAM. The implementation was done in the C language.



(a) Attribute Type $ATTRIBUTE\_1$ (b) Attribute Type $ATTRIBUTE = 1$

(c) Enc with Variable File Size (d) Dec with Variable File Size

(e) Enc with Variable Policy Size (f) Dec with Variable Policy Size

Fig. 3 Overview of the experimental results. **a** Attribute type *ATTRIBUTE*_1. **b** Attribute type *ATTRIBUTE* = 1. **c** Enc with variable file size. **d** Dec with variable file size. **e** Enc with variable policy size. **f** Dec with variable policy size

Furthermore, to provide a well-rounded evaluation of the protocol's performance we simulated a plethora of scenarios using different parameters. To acquire accurate measurements, we ran each experiment 50 times and calculated the average time needed to successfully complete the underlying process. Experiments to measure the execution times of functions related to the CSP and the communication channels utilized by the proposed protocol were considered to be out the scope of this experimentation section. The experiments were carried out in phases according to the RAP protocol steps.

Setup This phase was dedicated to generating the keys that are used for all cryptographic functions. This phase corresponds to the RAP.Setup in our proposed protocol. We measured the time needed to generate the master public/private key pair using CPABE.Setup as well as the RSA public/private key pairs for each entity. In our protocol a single MA exists—hence, a single (MPK, MSK) key pair is generated during the setup phase. The average execution time measured to generate the (MPK, MSK) key pair was 0.014 s in 50 iterations. Furthermore, the time required to generate each user's unique (pk, sk) key pair was measured at an average of 0.086 s per user in 50 iterations.

User key generation In these experiments we focused on the generation of an emergency CPABE key for each treatment team—a functionality that is part of protocol's RAP.GrantAccess phase. More precisely, we measured the processing time of the CPABE.Gen function that takes as input an arbitrary number of attributes 560 and outputs a unique secret key for each entity. Two types of attributes were used for these experiments. The first type is of the form ATTRIBUTE_i (i.e. a list of attributes), while the other is of the form ATTRIBUTE = i (i.e. we assigned values to attributes). The reason for using different types of attributes is that, while running our experiments, we identified significant differences in the processing time. More precisely, generating keys with attributes to which we have assigned values (e.g. ATTRIBUTE = i) required significantly more time for the generation of a key. The experiments involved an arbitrary number of attributes from 1 to 20. The results of the experiments varied greatly depending on the type of attributes used. For attributes of the type ATTRIBUTE_i, execution times measured for generating a CPABE key with 5 attributes was about 0.026 s, while with 20 attributes it was about 0.102 s in 50 iterations (Fig. 3a). For attributes of the type ATTRIBUTE = i, the execution times measured for generating a CPABE key with 5 attributes it was about 1.642 s and with 20 attributes was about 6.306 s in 50 iterations (Fig. 3b). Apart from that, generating CPABE keys increases linearly with the number of attributes for both types of attributes. Finally, by comparing the results shown in Fig. 3 a and b, we see that generating keys with simple attributes (i.e. no assigned values) will result in a more efficient implementation of our protocol.

EMR file encryption and decryption A core function of our protocol is the encryption and decryption of a patient's EMR file using CPABE. To measure this process, we ran experiments where we encrypted files with various sizes associated with an emergency policy and a set of attributes. The combination of an arbitrary number of attributes and file sizes allowed us to simulate more realistic cases. The first part of this experiment involved files of different sizes with a fixed number of attributes (i.e. static policy size). We encrypted files of sizes ranging from 1 to 20 MB with a policy requiring five attributes of type ATTRIBUTE_i. The file of size 1 MB was encrypted in about 0.034 s and decrypted in about 0.019 s. The file with size 20 MB was encrypted in about 0.1567 s and decrypted in about 0.1784 s. Figure 3 c and d show that the processing time increases linearly as the file size is increased.

The next experiment involved a file of fixed size and a policy with a variable number of attributes. We encrypted a file of 5 MB with a policy with 5 to 20 attributes of the same type as in the previous phase. Encryption and decryption times increased as the policy size increased. A file of size 5 MB with a policy of size 20 was encrypted in about 0.1337 s and successfully decrypted in about 0.0789 s. Figure 3 e and f illustrate the results of our experiments with a policy of variable size.

Token generation, signing and verification Our protocol depends heavily on the token generated by the MA in RAP.GrantAccess. In our experiments, we measured the time taken to generate the token, that is, to generate a message comprising of t_{gen}, t_{exp}, $\mathrm{Enc}_{\mathrm{pk}_{\mathrm{CSP}}}(r_2, s_{x1}, s_{x2}, u_i)$ and $\sigma_{\mathrm{MA}}(H_\tau)$. Our results indicate a total execution time of about 1.671×10^{-3} sec.

10 Discussion

From the results of the presented experiments, we confirm that the performance of the encryption and decryption functions depends on the size of the policy of the ciphertext, the attributes attached to a user's secret key and the size of the EMR file. The overall performance can be improved by optimizing the way we generate the attributes. Attributes of the type ATTRIBUTE_i should be utilized as the execution times for these are more efficient. Furthermore, it is evident from the experimental results that the time needed for the execution of the protocol renders our construction feasible, even when we increase the number of attributes. As a next step, we plan to experiment with different ABE schemes in order to find the

one that best suites our construction and evaluate the performance with larger size files, which would be more realistic for images and signals data.

In the protocol, we assume that all users are registered with the RA. However, we understand that there are cases when the patient is not registered or cannot be identified. Thus, for those cases, one possible option is to create a temporary 'John Doe' user to receive the record of the current stroke acute care patient. Using this approach, the professionals would still be able to use the system to share information about patient treatment. However, RAP needs to be able to merge the information as soon as the patient is identified. We plan to support those cases in the next version of the protocol.

In addition, the protocol relies on the token revocation list in the CSP to do the access control. The only way to bypass this is through an internal attack on the CSP. To strengthen the CSP, we could assume the existence of a trusted execution environment, such as Intel SGX [24], that will further secure the token-based access control. We believe that SGX is a good candidate for our construction since it offers isolation, sealing and attestation functionalities. More information can be found at [24, 25].

Moreover, it is important to emphasize that the access to the EMR must be implemented through a secure read-only application, where the EMR is decrypted by the professional using the CPABE emergency key. The application must not allow downloading the files. Thus, the EMR is just available during the emergency session.

11 Conclusion

In this paper, we proposed Red Alert, a protocol based on ciphertext-policy attribute-based encryption that allows access control to encrypted medical data during emergency situations. The proposed scheme enables healthcare professionals to decrypt a patient's encrypted data by making use of time-based tokens that are issued during the emergency situation. After the expiration of the tokens, the users are revoked and can no longer access the patient's data. The security of our scheme is proven using both simulation-based security as well as direct attacks on the protocol. Finally, we proved that the time for the RAP core functions execution is feasible in an emergency situation, since the approximate sum of execution times of the primary functions is below 0.5 s and the message exchange between the entities would happen before the patient be actually under the team treatment. Therefore, RAP enables the patient's EMR availability for the teams before the treatment begins, which can potentially improve patient care without compromising the security and patient's privacy.

Funding information This work was funded by the ASCLEPIOS: Advanced Secure Cloud Encrypted Platform for Internationally Orchestrated Solutions in Healthcare Project No. 826093 EU research project.

References

1. Hillestad R, Bigelow J, Bower A, Girosi F, Meili R, Scoville R, Taylor R (2005) Can electronic medical record systems transform health care? Potential health benefits, savings, and costs. Health Aff 24(5):1103–1117
2. Abbas A, Khan SU (2014) A review on the state-of-the-art privacy-preserving approaches in the e-health clouds. IEEE J Biomed Health Inform 18(4):1431–1441
3. Mashima D, Ahamad M (2012) Enhancing accountability of electronic health 660 record usage via patient-centric monitoring, in: Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium, ACM, pp 409–418
4. Zhang R, Liu L (2010) Security models and requirements for healthcare application clouds, in: 2010 IEEE 3rd International Conference on cloud Computing, IEEE, pp 268–275
5. Saver JL (2006) Time is brain-quantified. Stroke 37(1):263–266
6. Li M, Yu S, Ren K, Lou W (2010) Securing personal health records in cloud computing: patient-centric and fine-grained data access control in multi-owner settings. In: International conference on security and privacy in communication systems. Springer, pp 89–106
7. Brucker AD, Petritsch H, Weber SG (2010) Attribute-based encryption with break-glass. In: IFIP International Workshop on Information Security Theory and Practices. Springer, pp 237–244
8. Yang Y, Zheng X, Guo W, Liu X, Chang V (2019) Privacy-preserving smart iot-based healthcare big data storage and self-adaptive access control system. Inf Sci 479:567–592
9. Scafuro A (2019) Break-glass encryption. In: IACR International Workshop on Public Key Cryptography. Springer, pp 34–62
10. Povey D (1999) Optimistic security: a new access control paradigm. In: Proceedings of the 1999 workshop on New security paradigms. ACM, pp 40–45
11. Brucker AD, Petritsch H (2009) Extending access control models with break-glass. In: Proceedings of the 14th ACM Symposium on Access Control Models and Technologies. SACMAT '09, ACM, New York, pp 197–206
12. Zhang T, Chow SS, Sun J (2016) Password-controlled encryption with accountable break-glass access. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. ACM
13. Marinovic S, Craven R, Ma J, Dulay N (2011) Rumpole: a flexible break-glass access control model. In: Proceedings of the 16th ACM symposium on Access control models and technologies. ACM
14. Wallner D, Harder E, Agee R et al (1999) Key management for multicast: issues and architectures. Tech. rep., RFC 2627
15. Canetti R, Garay J, Itkis G, Micciancio D, Naor M, Pinkas B (1999) Multicast security: a taxonomy and some efficient constructions. In: IEEE INFOCOM' 99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE

Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320), vol 2. IEEE, pp 708–716
16. Rafaeli S, Hutchison D (2003) A survey of key management for secure group communication. ACM Comput Surv (CSUR) 35(3): 309–329
17. Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: Proceedings of the 2007 IEEE Symposium on Security and Privacy, SP'07. IEEE Computer Society, Washington, DC, pp 321–334. https://doi.org/10.1109/SP.2007.11
18. Michalas A (2016) Sharing in the rain: secure and efficient data sharing for the cloud, in: 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST). IEEE
19. Paladi N, Gehrmann C, Michalas A (2017) Providing user security guarantees in public infrastructure clouds. IEEE Trans Cloud Comput 5(3):405–419. https://doi.org/10.1109/TCC.2016.2525991
20. Dolev D, Yao AC On the security of public key protocols, Information Theory, IEEE Transactions on 29(2)
21. Michalas A (2019) The lord of the shares: combining attribute-based encryption and searchable encryption for flexible data sharing. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC '19. ACM, New York, pp 146–155. https://doi.org/10.1145/3297280.3297297
22. Oliveira MT, Michalas A, Groot AED, Marquering HA, Olabarriaga SD (2019) Red alert: break-glass protocol to access encrypted medical records in the cloud. In: HealthCom 2019-international conference on e-health networking, applications and services, IEEE. pp 1–7. https://doi.org/10.1109/HealthCom46333.2019.9009598
23. Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: Proceedings - S and P 2007. Proceedings - IEEE Symposium on Security and Privacy, pp 321–334. https://doi.org/10.1109/SP.2007.11
24. Bakas A, Michalas A (2019) Modern family: a revocable hybrid encryption scheme based on attribute-based encryption, symmetric searchable encryption and SGX. In: Chen S, Choo KK, Fu X, Lou W, Mohaisen A (eds) Security and privacy in communication networks. securecomm 2019. Lecture notes of the institute for computer sciences, social informatics and telecommunications engineering, vol 305. Springer, Cham. https://doi.org/10.1007/978-3-030-37231-6_28
25. Michalas A, Bakas A, Dang HV, Zalitko A (2019) MicroSCOPE: enabling access control in searchable encryption with the use of attribute-based encryption and SGX. In: Askarov A, Hansen R, Rafnsson W (eds) Secure IT systems. NordSec 2019. Lecture notes in computer science, vol 11875. Springer, Cham. https://doi.org/10.1007/978-3-030-35055-0_16