Towards Privacy-Preserving Access Control with Hidden Policies, Hidden Credentials and Hidden Decisions

Marian Harbach, Sascha Fahl, Michael Brenner, Thomas Muders and Matthew Smith
Distributed Computing and Security Group
Leibniz University Hannover
Schloßwender Str. 5, 30159 Hannover, Germany
Email: {harbach,fahl,brenner,muders,smith}@dcsec.uni-hannover.de

***Abstract*—The growing adoption of cloud technology in sensitive application domains, such as medicine, gives rise to new problems in maintaining the privacy of the involved parties during authorisation. In such domains, an honest but curious service provider can derive sensitive information purely from the authorisation process. In this paper, we present a detailed discussion of this rising problem including a concrete example and argue the need for the combination of hidden credentials, hidden policies and hidden decisions. We then show that mechanisms explored in previous work only cover individual aspects of this problem, but do not achieve a comprehensive solution without making restrictive assumptions on the resources, policies or subjects to be protected.**

As a first step towards solving this problem, we introduce an abstract foundation for using homomorphic cryptography to provide the required combination of privacy as a wrapper for other access control (AC) mechanisms. We achieve hidden policies, hidden credentials and even hidden access control decisions, so that the subject of an AC request only learns whether or not access was granted. Meanwhile, the provider of a resource learns nothing at the policy decision point and only access frequencies for individual resources at the policy enforcement point. We postulate that this is the maximum achievable level of protection in the authorisation process, without making restrictive assumptions on the resources, policies or subjects to be protected. Once homomorphic cryptography achieves satisfactory performance, our model can be used to transparently add this protection to other access control models.

***Index Terms*—Access Control, Homomorphic Cryptography, Privacy, Hidden Policies, Hidden Credentials, Hidden Decisions**

I. Introduction

The evolution of IT systems progressed from local, non-networked machines, through local machines with Internet access, to Service Oriented and Cloud Computing, where the actual location of a service provider is often of little interest. Consequently, security mechanisms evolved from locked doors through local digital measures to remote IT security mechanisms. Consequently, this trend of outsourcing IT services also causes the outsourcing of the safekeeping of that data. The economy of scale allows for a more effective use of resources, but also means giving a lot of information to a remote entity, hereby increasing data mining and profiling possibilities.

One important feature of IT security is access control (AC), which is primarily centred around deciding which services or resources can be accessed by which subjects (authorisation). There are several entities involved in an AC decision: A *subject* tries to access a *resource*, which is safeguarded by a *provider*. At the provider, there is a *policy enforcement point (PEP)*, that enforces whether or not a subject can proceed to access a chosen resource. The PEP delegates this decision to a *policy decision point (PDP)*, that uses a set of *credentials* (for example a digital identity or some digital attributes) to evaluate pre-defined *policies* and arrive at an access control decision.

When a human uses an IT system to access resources, a central issue is that of trust. The user needs to decide whether or not he can safely submit information to another party without the risk of unwanted disclosure. In the real world, intuition helps to assess trust towards that other party before and during an interaction. Because the contact of the two parties in a digital interaction is rather indirect, it is often impossible to assess mutual trust with the tools that evolution gave humans. Hence, there is a need to limit the initial disclosure of information at either party before trust is established to preserve their privacy. Therefore, Access control mechanisms are of central interest, because an AC request is often the first point of contact when beginning an interaction across IT systems.

During the authorisation process, several pieces of information potentially need to be protected, as was detailed in previous work (e. g. [1], [2], [3], [4]):

- the subject's credentials,
- which resource the subject requested,
- the provider's policies,
- the point in time when access was requested, and
- the decision that was actually taken.

For example, if a malicious or at least curious provider logs this information, a detailed profile can be derived from the requested resources (e. g. if one subject was requesting information on treatments for a specific illness – cf. [5]). Conversely, if a subject learned a provider's policies, the discovery of vulnerabilities or gaming the system can be facilitated. We will describe a scenario where partial information of this kind

is already sufficient to disclose personal information.

Furthermore, with digital repositories for information becoming more commonplace, it is paramount that users retain control over their data. Requiring administrators to configure rules or access control lists may no longer be an option in the future, when large numbers of users store and share digital artefacts. Hence, users need to be able to understand and configure AC policies for their information themselves (cf. [6], [7]). Consequently, there is a need for expressive, flexible and intuitive policy formalisms that can use a resource's context or metadata to make AC decisions. Such formalisms, however, can also raise additional security concerns. Disclosing such expressive policies in order to enable an AC decision (e. g. in [4]) could potentially allow the deduction of additional information, especially if the policies are sent to the subject to fulfil a request.

A lot of work has been put into methods to keep different parts of the above information private, while maintaining as much flexibility for policy definition and enforcement as possible. In related work, the hiding of credentials (Hidden Credentials [4]) or policies (Hidden Policies, [3]) or a combination of both has been described. However, the approaches in this area have serious inherent conceptual limitations, concerning resource types or policy expressivity. To the best of our knowledge, no approach in the literature prevents the subject from learning the policies and the provider from learning the credentials as well as the actual outcome of the decision process at the same time, while allowing expressive policies to be evaluated using an AC paradigm of choice. In line with Hidden Policies (HP) and Hidden Credentials (HC), we call preventing the provider from learning the outcome Hidden Decisions (HD). In this paper, we will argue for the need of all three properties in sensitive scenarios and present an approach for Homomorphic cryptography Supported Access Control (HSAC) as a first step towards achieving all three properties while imposing only minimal restrictions on the policies and resources.

Most state-of-the-art approaches achieve improved privacy at the cost of policy expressivity, communication overhead or restrictions on the type of resources. In contrast, the properties of HSAC allow for the combination with existing AC paradigms, such as discretionary [8], mandatory [9], role-based [10] or attribute-based [11] access control mechanisms and their variants.

Using HSAC, we can achieve a very high level of privacy for access control decisions. We will show how the subject's credentials, the provider's policies and the AC decision can be effectively hidden. Overall, the subject only learns whether or not access was granted and the provider does not learn anything about the subject or his own decision process. The only information a provider can gather are resource access frequencies, without making assumptions on the type of resource to be protected. We will also outline additional measures that can be taken to further reduce the amount of information that can be deduced from access logs.

The remainder of this paper is structured as follows. The next section presents a scenario to detail the need for privacy in sensitive access control applications, using an example from the much discussed area of Electronic Health Records (EHRs) in the cloud. Section III outlines relevant related work to this and similar problems. Section IV will subsequently provide a description of the HSAC model as well as a security analysis. Section V describes the limitations pertaining to this approach and points out directions for future work, before Section VI concludes this paper.

II. Scenario

The following section will outline a sample scenario in the medical domain that illustrates the need for an access control system which combines hidden policies, hidden credentials and hidden decisions.

A. Accessing Medical Data

In the medical domain, digital patient information is stored in Electronic or Personal Health Records (EHRs or PHRs). An EHR typically consists of payload information, such as x-ray images or blood test results, as well as meta-information about the patient. Such meta-information might comprise the patient's name, age, sex and address. Standards such as HL7 [12] or DICOM [13] propose protocols for the standardised exchange of information between care providers and EHRs for different areas of application (e. g. radiology or paediatricians). In medical infrastructures, EHRs are stored in Health Information Systems (HIS). Each access request of a medical staff member is typically issued by interaction with the HIS. Such a request consists of a source (e. g. the workstation of the medical staff or a user ID) and the identity of the patient to find the required data for.

With the adoption of IT, medical records were no longer exclusively managed in a paper-based fashion, but stored in a HIS. In a conventional setup, a HIS is located in a medical institution and requests for medical data are initiated by local medical staff only. Cloud computing, as a new paradigm in the IT world in general and for medical applications in particular, brought in a new concept for storing and processing medical records. In addition to local and centralised infrastructures for storing EHRs, there are solutions, such as Microsoft's Health Vault[1], that act as a central repository for medical data. Pushing medical information into the cloud brings a number of advantages:

1) Patients and physicians can access the information whenever and wherever needed (e. g. at the doctor's workstations or the patient's smartphone).
2) Patients can be enabled to govern access to the stored medical information.
3) Medical information can be more easily made available to the necessary entities.
4) In case of an emergency, an emergency doctor can immediately access all required information and hence treat the patient as required.

[1] See http://www.microsoft.com/healthvault/

In addition to advantages by storing medical records in a central repository, several drawbacks from a security and privacy point of view exist (cf. Löhr et al. [14]). Technically, security and privacy concerns may arise at the central storage repository as well as at the medical staff's workstation. Löhr et al. describe access control mechanisms as well as confidentiality of medical records as two major challenges for IT security concepts. They list some security problems, such as medical institutions' workstation security, mobile storage devices and the management of the e-Health infrastructure in general. Hereafter, we focus on access control and confidentiality issues.

The appropriate application of data encryption can successfully prevent a repository provider from reading the content of patients' medical records. Hence, only the encrypted version of a medical record is stored in the cloud and transferred to the device (e. g. a hospital's workstation) to display the record in cleartext. The decryption procedure is applied locally at the displaying device. To protect medical records from unauthorised access, appropriate access control mechanisms can be applied and therefore, only valid access requests are granted. The security model described above is usually applied to store and protect EHRs in an external repository.

In the following, we describe a threat for the patient's privacy, even in case of appropriate confidentiality as well as access control mechanisms being in place. Each access request consists of some inherent properties which might allow an honest but curious repository provider to obtain information about a patient's disease pattern, even if the medical records are encrypted. Those properties are the following:

1) To grant access to a medical record, the repository provider needs the identity or the credentials of the specific subject. Only with information about the identity or credentials, access control policies can be evaluated.
2) Since the transfer of medical records from a central repository to the respective medical institution takes place on computer networks, a repository provider can learn some information about the requestor, e. g. the source IP address or a X.509 domain name – in case TLS is applied for transport security – of the medical institution.
3) After evaluating the request, the repository provider also learns which policy authorised the request and whether or not the request was successful.
4) Additionally, a temporal pattern of access potentially discloses information on the treatment or disease that a subject is treated for.

As a result, a curious repository provider is able to gather information about the patient whose medical records are requested and about the institution from which the request was sent. With the knowledge of the source of an access control request, the repository provider might be able to isolate the type of disease the patient suffers from. A combination of the request's origin, i. e. the requesting physician or institution, and a temporal pattern can additionally allow further deductions. For example, if a patient's medical record is requested by a renal specialist, this fact is indicative of the patient being treated for kidney problems. If these requests occur e. g. three times a week, this might indicate that this patient is in constant need of dialysis.

In systems which aim at protecting the requestor's credentials, policies or parts thereof are often disclosed to the requestor in order to enable this (cf. e. g. [15], [16]). While the protection of the credential is a privacy benefit in our scenario, the disclosure of the policies can give rise to new privacy threats. For instance, consider a company physician doing a routine checkup on an office clerk's eyes, for which he is granted access to documents from the clerk's EHR. If the EHR repository's policy were to be disclosed to the company physician, as can happen with hidden credential AC systems, he might see a policy authorising a dialysis centre, thus learning that this particular person suffers from a renal disease. Therefore, in this scenario, hiding the policies from the subject is required as well.

III. Related Work

There are several approaches to hide information during the authorisation process in similar scenarios. However, considering the scenario above, there are still some unsolved problems and we will discuss existing approaches in the following.

Holt et al. [4] described the original Hidden Credentials system, using IBE to encrypt messages. The subject can only decrypt a message (the resource to be accessed) if he possesses the credential. Holt et al. use <*username*||*attribute*> as the IBE public key and therefore allow the encoding of policies as a combination of cryptographic operations on the resource to be shared. One problem of this approach is that the secret message is sent across the network anyway, which might not be desirable for very sensitive or large amounts of data. Additionally, this scheme allows for offline probing and will therefore allow an adversary to game the system. Removing one user's access privileges requires re-encryption of the message.

Backes et al. [17] achieve privacy-enhanced access control using zero knowledge proofs while providing accountability. However, their method discloses policies and allows the provider to learn the access decision.

Camenish et al. [2] describe the importance and requirements for subject privacy in an electronic society and propose an infrastructure for privacy preservation according to EU legislation. Access control is managed using anonymous credentials certified by a third party and mutual trust achieved using negotiation mechanisms, but neither policies nor credentials remain hidden in this system.

Frikken et al. [3] propose a first approach that achieves hidden policies and hidden credentials for attribute based authorisation. In their most secure protocol, the provider also does not learn the decision that was taken. However, the scheme relies heavily on the exchange of information that can potentially leak some information. Additionally, with a growing number of attributes in the system, the communication

complexity increases exponentially and in general, the system works only for policies that check for the presence of certain attributes.

In a 2011 follow up (cf. [1]), Camenish et al. propose an AC mechanism that provides very strong security properties, similar to the approach described in this paper, providing hidden credentials, hidden policies and anonymous access. However, their model mainly applies to read-only databases that can be distributed to all participants, for example by mailing DVDs or providing a download for the entire database. Therefore, while being able to implement and provide the necessary security properties, the assumptions on the nature of the data and the mode of distribution allow for very little flexibility in terms of resource types and access modes.

Another line of research focuses on minimising the amount of information that is disclosed until sufficient trust is established. For example Ardagna et al. [15] and other approaches for trust negotiation (e. g. [16]) use policies that regulate which information is disclosed under which circumstances. This approach only limits the amount of information being disclosed but cannot prevent the provider from learning which resources were accessed or which credentials were used for a successful request.

Vaidya et al. [18] propose a system that allows the subject to specify a preference on which credentials to disclose and also describe three methods for arriving at an access control decision in a privacy-preserving manner. They assume that a requestor will eventually be willing to disclose a necessary subset of attributes in order to gain access.

The approach of Yao et al. ([19], [20]) suggests using a systems of points for selecting a minimal amount of credentials to disclose, based on a threshold required for being granted access by the provider and a 'privacy-cost" for each credential chosen by the requestor. The method again suggests that a requestor is willing to disclose the necessary set of credentials eventually.

In order to protect sensitive attributes from disclosure, Lin et al. [21] propose splitting global policies across a federation of PDPs. Departments of the same company, for example, might not be willing to have a global PDP see sensitive attribute values necessary for a policy decision. They use algorithms to delegate the decisions to individual PDPs and integrate the individual decisions. Obviously, this approach is only useful within one organisational domain, since the policies need to be engineered accordingly, and does not easily lend itself to hide the subject's credentials.

Finally, Sedghi et al. [5] argue that Hidden Credentials (HC) and Attribute-Based Encryption (ABE) as well as Predicate Encryption (PE) schemes do not suffice to prevent an honest but curious reference monitor (a PEP in their work) from learning information that either the subject or the provider want to remain private. They propose a model that enables a server (the provider) to enforce a policy without disclosing policies nor learning a subject's attributes/credentials, using a trusted third party, that holds the subject-to-role mapping and has the ability to translate blinded roles and credentials.

Overall, most AC methods proposed in related work impose restrictions on the policy formalism or the type of resource or access-type that can be used. Systems with less restrictions on the components usually assume that the involved parties are generally willing to disclose information to a trusted party and mainly deal with establishing that trust during the AC process. Altogether, to the best of our knowledge, there are no AC mechanisms that can hide policies, credentials and decisions completely and hence maximise privacy for the involved parties.

IV. HSAC: A Universal Approach

In order to achieve the desired functionality described above, we propose to make anonymous and oblivious access control decisions and grant access to resources based on tickets, using a universal homomorphic computation container (UHCC). The UHCC will allow for the integration of a suitable AC mechanism, such as a simple AC list, RBAC, or many other existing methods. We combine this to create the foundation for Homomorphic cryptography Supported Access Control (HSAC). Besides few assumptions about the infrastructure, this foundation allows the system to be tailored to the requirements of its application while achieving a maximum of privacy.

We assume that the resource provider and the subject follow the push sequence for getting access to a resource, as described in RFC 2904. A subject obtains an authorisation ticket from an organisation's AAA Server (the PDP) and presents this ticket to the service equipment (the PEP), therefore proving that access was allowed by the PDP (cf. Figure 1). This indirection allows for maximised privacy during the PDP's decision process. Using the push sequence, the PEP cannot know when or by whom the ticket was obtained. Compared to the more direct pull sequence (the PEP obtains the authorisation from PDP itself), the push sequence allows for the separation of the PDP and PEP and only increases communication complexity from one to a second exchange.

This is desirable if the subject has an interest in maintaing his privacy and therefore in protecting the ticket[2] and has the advantage of separation between PDP and PEP. However, PDP and PEP will usually reside within the same organisation, and hence an honest but curious provider can correlate information from both entities. We will demonstrate in Section IV-C though, that, based on this theoretical foundation, there is nothing to be learned from a possible collusion between PDP and PEP, except access frequencies for individual resources. Meanwhile, the PDP as well as the PEP do not learn anything about the resource's content, the request's origin or the requestor's identity.

The following subsection will briefly outline an UHCC's functionality and basic properties as proposed by Brenner et al. (cf. [22], [23]), before we describe its application for anonymous access control.

[2]Obviously, this behaviour is not useful in scenarios where a resource provider does not want the subject to pass a ticket on to another subject (e. g. at an online movie rental site).

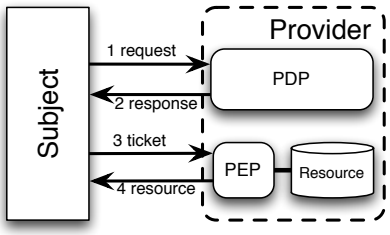

Fig. 1. Push-Sequence for AAA, adapted from RFC 2904.

A. Homomorphic Container

To evaluate a function f in encrypted space, the container applies an abstraction of the fully homomorphic Smart-Vercauteren cryptosystem [24]. f is therefore encoded into an encrypted program for a runtime container which, in turn, implements a general purpose CPU with random access memory over homomorphically encrypted boolean circuits (cf. [22] and Fig. 2). Basically, the UHCC is a fully-fledged computer, operating on encrypted inputs using encrypted programs, and producing encrypted outputs.

The encrypted program (i. e. the implementation of f) is assembled and encrypted into a memory image by an owner A and then secretly evaluated at an executing entity B, running an encrypted virtual machine. After execution, parts of the modified memory image, containing the encrypted outcome of the function evaluation, are transferred back to A. A decrypts the image and hence obtains a secretly computed result.

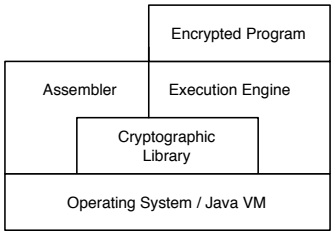

Fig. 2. Architecture of the UHCC [25].

The programming model for the encrypted CPU requires the machine program to be encrypted at assembly time using a public key of A, which is available in combination with the memory image to B. The result can be decrypted with A's private key. The machine program uses a basic command set, which is sufficient to model arbitrary functions [23]. The homomorphic scheme allows for injection of data into the memory image – even after it was transferred to B. To achieve this, plaintext is encrypted using A's public key and the ciphers are then inserted into designated memory locations, i. e. locations where the implementation of f expects external input parameters.

B. The HSAC Model

Our access control model works with the following abstraction: We assume that there is a function

$$f : \wp(S) \times R \times \wp(P) \times \wp(E) \rightarrow \{0, 1\}$$

that decides whether or not a subject with a set of subject credentials $s = \{s_1, \ldots, s_l\} \subset S$ has access to a resource $r \in R$, considering policies $p = \{p_1, \ldots, p_m\} \subset P$ and a set of static environment properties $e = \{e_1, \ldots, e_n\} \subset E$. Here, f represents the abstract access control model used to make an AC decision. Note that f is an arbitrary function, meaning that it might well be non-linear.

As a simple example, using 'vanilla" RBAC, S would only consist of a subject identification, while P would model subject-to-role, role-to-permission and resource-to-permission mappings, and E is empty. f returns 1 if there is a corresponding mapping for a subject id to the requested resource. We do not need to explicitly model actions on resources as a parameter of f, since these can be easily represented using extended resource descriptors (e. g. r_w for write access on resource r).

As mentioned above, after arriving at an access control decision, the PDP issues a ticket that the subject can use to access the corresponding resource at the PEP (the push sequence from RFC 2904 outlined above). In order to authenticate the ticket, the PDP signs the tickets with its private key (k_{priv}^{PDP}), which the PEP and the subject can verify using the corresponding public key (k_{pub}^{PDP}). k_{pub}^{PDP} can be assumed to be certified using available certificate authorities and hence be freely distributed and easily authenticated by the PEP and the subject. Additionally, the subject has a public/private key pair ($k_{pub}^{subject}$,$k_{priv}^{subject}$) used for the encryption and decryption of the UHCC.

There is a risk of reidentification of subjects, using their public UHCC keys. We propose to randomly choose one of multiple key pairs for consecutive requests to impede pattern extraction based on this information. Therefore, for each request, the subject randomly chooses $i : 0 \leq i < n$, where n is the total number of key pairs, which needs to be larger the more requests are sent by this particular subject. Public key distribution is not a concern, since the keys can be sent along with each request. The subject then selects $(k_{pub}^{subject}, k_{priv}^{subject})_i$ for the exchange with the provider. In the following, we will omit the key pair index i for brevity.

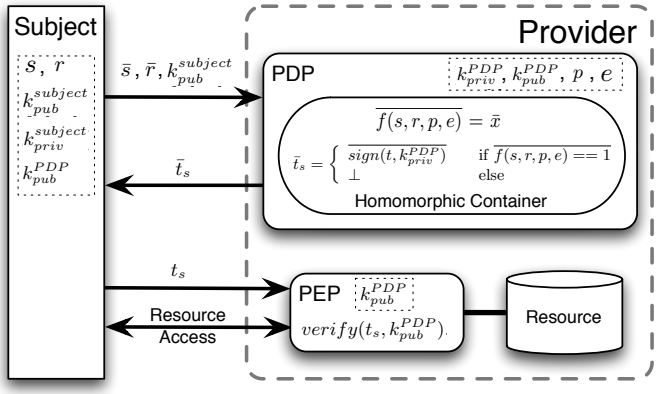

Fig. 3. Information exchange in the HSAC model.

The crucial part is the computation of f at the PDP, where the provider must not be able to learn f's result and the parameters provided by the subject. Therefore, we wrap f into a homomorphic container that is encrypted with $k_{pub}^{subject}$. Therefore, the subject encrypts its credentials s and the desired resource identifier r with $k_{pub}^{subject}$ into the homomorphic cipherspace – creating $\bar{s}$ and $\bar{r}$ – and sends these together with $k_{pub}^{subject}$ to the PDP. If the subject wants to prevent the provider from learning the number of credentials presented, the list of credentials s can be padded before encryption (for example with multiple copies of the same credential). The PDP uses the subject's public key to encrypt the algorithm implementing f as well as its inputs p and e – creating $\bar{f}$, $\bar{p}$ and $\bar{e}$ – for evaluation with $\bar{s}$ and $\bar{r}$ in cipherspace. Eventually, inside the container, an encrypted and signed ticket $\bar{t}_s$ is created in cipherspace:

$$\bar{t}_s = \begin{cases} \overline{sign(t, k_{priv}^{PDP})} & \text{if } \overline{f(s,r,p,e) == 1} \\ \bot & \text{else} \end{cases}$$

and placed at a predetermined location in the encrypted memory. Note that, because the provider effectively computes $\overline{f(s,r,p,e)} = \bar{x}$ and uses an encrypted version of the equality operator to compare the result $\bar{x}$ with an encrypted representation of 1, he cannot learn anything about this evaluation. To authenticate the ticket, t was also signed with the PDP's private key k_{priv}^{PDP} in cipherspace, which is injected before execution as well.

After the computation inside the container is done, the result is extracted from a predetermined location in the encrypted memory and sent back to the subject. On obtaining the result, the subject is able to decrypt the result using $k_{priv}^{subject}$. If the subject finds a valid ticket t_s that is verifiable using k_{pub}^{PDP}, he can then use t_s to convince a PEP of his access permission. Again, the PEP can validate the PDP's signature, using $verify(t_s, k_{pub}^{PDP})$. Figure 3 provides an overview of the exchange between subject and provider.

Additionally, two remaining information leaks need to be addressed:

1) Network Anonymisation: To prevent the provider from learning anything about the requestor using network layer information, the request can be transferred through onion networks, such as Tor, or other anonymisation technologies (e. g. [26], [27]). This measure is less important if the subject uses dynamic IP address allocation. However, if an institution has a dedicated IP address range, the provider can deduce information from that, for example that this request came from a hospital specialising in plastic surgery. Additionally, if multiple requests occur within the lifetime of the same IP address, it might be possible to learn more information about the subject across multiple requests. Especially if the provider supplies further services to the same subject, an IP address allows correlation of data and facilitates reidentification.

After the subject has extracted a valid ticket from the PDP's response, the ticket needs to be protected from interception because it would enable an arbitrary adversary to gain access to the resource instead of the subject. Hence, we propose to authenticate the PEP using for example a TLS channel and server authentication. The necessary certificates can be exchanged using commonplace methods. The TLS channel could also be used for the actual resource access after the ticket was accepted. At this point, it is important not to apply TLS client authentication, since this can reveal a subject's identity. Yet, this is not a security concern, since the actual AC mechanism will still protect the resources.

2) Resource Identification: It might seem that resource identifiers should and can be hidden from the provider, especially the PEP. This would prevent any information on a resource's content and purpose to be disclosed to the provider. In particular, this would mean that no metadata or structured identifiers could be used, which for example prevents the adoption of expressive access control mechanisms. However, there are some issues to be considered.

Generally, the PDP needs policies to base its decisions on. This means that someone needs to be able to configure the policies and effectively make statements like "Subject X is allowed to access Resource Y" using some formalism. Whatever formalism that is, due to the nature of access control, it can be broken down to an access control list (ACL) in finite time (given that the sets of subjects or credentials and resources are finite). Hence, if the PEP (i. e. the resource) and PDP collude (which has to be assumed, since they usually belong to the same organisation), a resource can be linked to some AC credentials through the resource identifier necessary to configure the policies. In the worst case, this association links the resource to certain subject identities and the implications of their access (e. g. access of some medical specialist can imply specific health problems). Using suitable policy formalisms, the extraction of information from the given set of policies can be impeded. The discussion of this topic is however out of scope for this paper.

Due to the application of the push sequence (cf. Fig. 1 above), the PDP can also be separated from the PEP. All the PEP needs is to trust the PDP's signature on the ticket. Hence, it is conceivable to locate the PDP and the PEP with the resource in different organisations, reducing the risk of collusion. If there is a trusted ticket-issuing organisation, that does not learn anything about the subjects obtaining tickets, but can provide a very expressive means to configure access control policies, then it is hard for a resource provider, that simply accepts those tickets and provides access to the corresponding resources, to correlate access frequencies to any other information. This separation of concerns maximises the anonymity in resource access control, but is only useful if there is a strong motivation to avoid collusion for both, PDP and PEP.

So, without making further assumptions, the PEP will in general be able to learn that some resource was accessed at specific points in time. In collusion with the PDP, the given set of policies can allow the deduction of further information on the resources, if the policy formalism lends itself for such a task. However, neither PDP or PEP will be able to deduce for whom and because of which policies access was granted, due

to the Hidden Decisions and Hidden Credentials properties of our approach.

C. Security Discussion

The process of requesting a resource from a provider offers multiple attack vectors for internal as well as external adversaries. Apart from common network-based attacks (e. g., DoS) and host-based attacks (e. g., malware), we will consider only direct attacks to the exchanges described in our framework. Additionally, a malicious PEP or resource is not within the scope of this paper, as securing resource contents themselves in a flexible way is an area of extensive research (see for example [28]).

First, an adversary is neither able to learn the credentials nor the desired resource from intercepting traffic between subject and provider. Both, $\bar{s}$ and $\bar{r}$ are encrypted prior to being transferred and can only be decrypted using $k_{priv}^{subject}$, which is kept only locally on the subject's host. Therefore, the two request parameters and the subject's public key $k_{pub}^{subject}$ might as well be transferred in clear. The adversary might learn the public key from listening to the exchange and could manipulate the provider's responses by injecting modified, but validly encrypted tickets. However, after extracting the ticket from $\bar{t}_s$, the PDP's signature would not validate and there is no way for the adversary to forge the signature without learning the PDP's private key k_{priv}^{PDP}. In worst case, this then leads to a denial of service, which cannot be avoided anyway, if the attacker is capable of arbitrarily modifying traffic.

Receiving $\bar{s}$, the provider might be able to deduce the number of credentials based on the size of the ciphertext. However, as mentioned above, if this is an information to be protected, the subject can choose to pad the list of credentials before encryption or choose not to disclose certain credentials in a request. Hence, the provider can possibly estimate an upper bound of the number of a subject's credentials. However, because there is only negligible probability for reidentification (using anonymous network access and rotating public keys), this information cannot be assigned to any specific subject.

While we assume the homomorphic container to be secure, based on the properties of the Smart-Vercauteren scheme (cf. [24]), we need to make sure that no information besides the signed ticket is transferred back to the subject. In case other parts of the container's memory image were transferred back to the subject, these parts could be decrypted and the plaintext contents analysed (e. g. the PDP's private key recovered). Our container however allows to extract predetermined parts from the image, so that only the ticket is specifically extracted and transferred. Therefore, the subject cannot learn the policies, the PDP's private key or even the AC mechanism that was used.

As mentioned above, the decrypted ticket needs to be protected by the subject. Besides securing the host and using a secure channel like TLS, additional measures can be taken to reduce the impact of ticket disclosure. If the ticket is invalidated after first use, the subject will at least find out that his ticket was stolen. Otherwise, a narrow validity window using a timestamp can also reduce the impact an adversary or a malicious subject may cause. At any rate, the PDP's signature ensures that only the resource authorised by the active policies can be accessed, given that the PEP enforces the information provided in the ticket correctly.

V. Limitations and Future Work

Most related systems having the desired properties impose restrictions on the policy structure, the representation of a subject's attributes, or the resource to be instrumented. Our work focused on the discussion of a more universal foundation for maximising AC privacy. However, currently there are still limitations that need to be addressed in future work. Yet, these limitations primarily concern the homomorphic computing environment and are therefore not inherent problems of our HSAC foundation.

First, the computational complexity of the Smart-Vercauteren cryptosystem [24] is very disadvantageous, due to the use of large integers to represent single bits. Currently, the calculation of a simple integer addition in the homomorphic container will take a few minutes to compute. Thus, the UHCC has considerable runtime restrictions, which limit its use for real world applications [22]. Yet, all our work is completely independent from the underlying homomorphic cryptography scheme. Thus, when faster homomorphic cryptosystems become available, they can be integrated without any conceptual implications on our HSAC approach. There is also ongoing work to provide a UHCC implementation in hardware, which will help to improve the runtime characteristics. Since our AC model allows for a physical and logical separation of the PDP and the PEP, specialised hardware can be easily fitted into a corresponding infrastructure. Brenner et al. discuss the practicality of homomorphically protected program execution in two recent papers [29], [30].

The anonymity feature of our approach can also be a limitation. As argued above, there are scenarios where detailed application-level logging is not desirable and can cause privacy problems. We therefore explicitly designed our framework in a way that no uncontrolled logging of the subject's information is possible. However, in cases where some form of audit is necessary, the provider could send a signed log proposal from within the homomorphic container along with the ticket back to the subject, who after decryption can inspect the entry and return it to the provider if he allows the log entry to be made. Another option is a trusted third party acting as a proxy that keeps the logs and handles the cryptography in a transparent way for the subject.

Altogether, this approach presents a foundation for researching fully privacy preserving evaluation of access control policies that has stronger privacy properties compared to current state-of-the-art proposals. Advances in the performance of fully or hybrid homomorphic cryptography will seamlessly improve the performance of this proposal.

VI. Conclusion

In this paper, we discussed the privacy needs of access control systems, based on available approaches and a scenario from cloud-based medical IT systems. As a first step towards this goal, we presented a theoretical foundation named Homomorphic cryptography Supported Access Control (HSAC) that allows for a very high level of privacy during the authorisation process. While other mechanisms impose conceptual limitations on the AC capabilities or the resources to be protected, this proposal has few restrictions, low communication cost and does not make tradeoffs between different protection goals.

We argued that unless restricting assumptions are made on the type of resources and their access patterns, the resource provider will always learn access frequencies. However, our approach also allows for the organisational separation of PDP and PEP, which can further improve a subject's privacy, if the two organisations have incentives to avoid collusion. Additionally, we argued that care needs to be taken when integrating privacy-preserving AC mechanisms in larger environments, as correlations between accesses can enable re-identification and hence profiling of subjects.

We also discussed the present limitations of our approach as well as directions for future research to overcome these. We believe that a universal approach with little restrictions on the AC model and the resources will help to make the use of information technology in general and availability of data in particular more comfortable and more secure at the same time.

References

[1] J. Camenisch, M. Dubovitskaya, G. Neven, and G. Zaverucha, "Oblivious Transfer with Hidden Access Control Policies," in *Public Key Cryptography*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2011, vol. 6571, pp. 192–209.

[2] J. Camenisch, A. Shelat, D. Sommer, S. Fischer-Hübner, M. Hansen, H. Krasemann, G. Lacoste, R. Leenes, and J. Tseng, "Privacy and Identity Management for Everyone," in *Proceedings of the 2005 Workshop on Digital Identity Management*. ACM, 2005, pp. 20–27.

[3] K. Frikken, M. Atallah, and J. Li, "Attribute-Based Access Control with Hidden Policies and Hidden Credentials," *IEEE Transactions on Computers*, vol. 55, no. 10, pp. 1259 –1270, oct. 2006.

[4] J. E. Holt, R. W. Bradshaw, K. E. Seamons, and H. Orman, "Hidden Credentials," in *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*. ACM, 2003, pp. 1–8.

[5] S. Sedghi, P. Hartel, W. Jonker, and S. Nikova, "Privacy Enhanced Access Control by Means of Policy Blinding," in *Information Security Practice and Experience*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2011, vol. 6672, pp. 108–122.

[6] M. Smith, M. Harbach, S. Mertins, A. Lewis, and L. Griffiths, "Towards a Translational Medicial Research Ecosystem," in *Proceedings of the 5th IEEE International Conference on Digital Ecosystems and Technologies*, 2011, pp. 120–126.

[7] M. Harbach and M. Smith, "Visual Access Control for Research Ecosystems," in *Proceedings of 5th IEEE International Conference on Digital Ecosystems and Technologies*, 2011, pp. 101–108.

[8] United States Department of Defense, "Trusted Computer System Evaluation Criteria (DoD 5200.28-STD)," United States Department of Defense, Dec 1985.

[9] D. E. Bell and L. J. LaPadula, "Secure Computer Systems: Mathematical Foundations and Model," The MITRE Corporation Bedford MA, Tech. Rep. M74-244, 1973.

[10] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," *Computer*, vol. 29, pp. 38–47, February 1996.

[11] E. Yuan and J. Tong, "Attributed based access control (ABAC) for Web services," in *IEEE International Conference on Web Services*, July 2005, p. 569.

[12] *HL7 Version 3 Standard: Clinical Document Architecture (CDA)*, Std., 2005.

[13] *DICOM Structured Reporting: Part 2. Problems and Challenges in Implementation for PACS Workstations*, Std. 3, 2004.

[14] H. Löhr, A.-R. Sadeghi, and M. Winandy, "Securing the e-Health Cloud," in *Proceedings of the 1st ACM International Health Informatics Symposium*. ACM, 2010, pp. 220–229.

[15] C. Ardagna, S. De Capitani di Vimercati, S. Foresti, G. Neven, S. Paraboschi, F.-S. Preiss, P. Samarati, and M. Verdicchio, "Fine-Grained Disclosure of Access Policies," in *Information and Communications Security*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, vol. 6476, pp. 16–30.

[16] F. Abel, J. L. Coi, N. Henze, A. W. Koesling, D. Krause, and D. Olmedilla, "The RDF Protune Policy Editor: Enabling Users to Protect Data in the Semantic Web," in *Web Information Systems and Technologies*, ser. Lecture Notes in Business Information Processing. Springer, 2010, vol. 45, pp. 142–156.

[17] M. Backes, J. Camenisch, and D. Sommer, "Anonymous yet Accountable Access Control," in *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*. ACM, 2005, pp. 40–46.

[18] J. Vaidya, V. Atluri, B. Shafiq, and N. Adam, "Privacy-preserving Trust Verification," in *Proceeding of the 15th ACM Symposium on Access Control Models and Technologies*. ACM, 2010, pp. 139–148.

[19] D. Yao, K. Frikken, M. Atallah, and R. Tamassia, "Point-Based Trust: Define How Much Privacy Is Worth," in *Information and Communications Security*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 4307, pp. 190–209.

[20] D. Yao, K. B. Frikken, M. J. Atallah, and R. Tamassia, "Private Information: To Reveal or not to Reveal," *ACM Trans. Inf. Syst. Secur.*, vol. 12, pp. 6:1–6:27, October 2008.

[21] D. Lin, P. Rao, E. Bertino, N. Li, and J. Lobo, "Policy Decomposition for Collaborative Access Control," in *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*. ACM, 2008, pp. 103–112.

[22] H. Perl, M. Brenner, and M. Smith, "An Implementation of the Fully Homomorphic Smart-Vercauteren Crypto-System (Poster)," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, 2011.

[23] M. Brenner, J. Wiebelitz, G. von Voigt, and M. Smith, "A Smart-Gentry Based Software System for Secret Program Execution," in *Proceedings of the 6th International Conference Security and Cryptography (SECRYPT)*. SciTePress, 2011.

[24] N. Smart and F. Vercauteren, "Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes," in *Public Key Cryptography*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, vol. 6056, pp. 420–443.

[25] M. Brenner, J. Wiebelitz, G. von Voigt, and M. Smith, "Secret Program Execution in the Cloud Applying Homomorphic Encryption," in *Proceedings of the 5th IEEE International Conference on Digital Ecosystems*. IEEE, 2011.

[26] O. Berthold, H. Federrath, and S. Köpsell, "Web MIXes: A System for Anonymous and Unobservable Internet Access," in *Designing Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2001, vol. 2009, pp. 115–129.

[27] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, "P^5: A Protocol For Scalable Anonymous Communication," in *IEEE Symposium on Security and Privacy*, 2002, pp. 58 – 70.

[28] S. Kamara and K. Lauter, "Cryptographic Cloud Storage," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, vol. 6054, pp. 136–149.

[29] M. Brenner, H. Perl, and M. Smith, "How Practical is Homomorphically Encrypted Program Execution? An Implementation and Performance Evaluation," in *Proceedings of the 11th IEEE TrustCom*, 2012, to appear.

[30] ——, "Practical Applications of Homomorphic Encryption," in *Proceedings of the 7th International Conference on Security and Cryptography*, 2012, to appear.