Privacy-Preserving Audit for Broker-Based Health Information Exchange

Se Eun Oh
University of Illinois at Urbana-Champaign
seeunoh2@illinois.edu

Ji Young Chun
University of Illinois at Urbana-Champaign
jychun@illinois.edu

Limin Jia
Carnegie Mellon University
liminjia@cmu.edu

Deepak Garg
Max Planck Institute for Software Systems
dg@mpi-sws.org

Carl A. Gunter
University of Illinois at Urbana-Champaign
cgunter@illinois.edu

Anupam Datta
Carnegie Mellon University
danupam@cmu.edu

ABSTRACT

Developments in health information technology have encouraged the establishment of distributed systems known as Health Information Exchanges (HIEs) to enable the sharing of patient records between institutions. In many cases, the parties running these exchanges wish to limit the amount of information they are responsible for holding because of sensitivities about patient information. Hence, there is an interest in broker-based HIEs that keep limited information in the exchange repositories. However, it is essential to audit these exchanges carefully due to risks of inappropriate data sharing. In this paper, we consider some of the requirements and present a design for auditing broker-based HIEs in a way that controls the information available in audit logs and regulates their release for investigations. Our approach is based on formal rules for audit and the use of Hierarchical Identity-Based Encryption (HIBE) to support staged release of data needed in audits and a balance between automated and manual reviews. We test our methodology via an extension of a standard for auditing HIEs called the Audit Trail and Node Authentication Profile (ATNA) protocol.

Categories and Subject Descriptors

F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Temporal logic*; D.4.6 [**Security and Protection**]: Cryptographic controls; K.4.1 [**Public Policy Issues**]: Privacy

Keywords

Audit; Health information technology; Formal logic; Hierarchical identity based encryption

CODASPY'14, March 03 - 05 2014, San Antonio, TX, USA

http://dx.doi.org/10.1145/2557547.2557576.

1. INTRODUCTION

Broker systems enable organizations to index and share data securely by providing key infrastructure services. Such brokers provide an alternative to central repositories in which data is held as well as indexed by a central authority. The broker provides some level of indexing, provides for the transport of data, and supports security functions like authentication and audit, often without holding much of the key data itself except during its transport. These systems are of growing interest in the healthcare sector where efforts are being made internationally to take advantage of Electronic Health Records (EHRs), which enable institutions to represent medical data in standardized formats suited to inter-domain sharing. An advantage of the broker solution is that it limits the amount of information the broker needs to hold thus limiting risk and liability. For example, in the U.S. there are many nascent state-sponsored Health Information Exchange (HIE) systems where the state is reluctant to hold EMR data themselves because citizens have concerns about sharing their health data with government. Broker-based exchanges mitigate this concern while providing key benefits like the ability of healthcare providers to assemble patients' medical records quickly at the point of care.

Broker-based HIEs face a special problem with their access audit which aims to detect illegitimate accesses as part of general ongoing monitoring or specific investigations. The more audit data the HIE keeps the better it supports this function, but the more risk there is that the audit information itself will compromise privacy. Moreover, large numbers of transactions make it impossible to carry out manual audits at scale so there is a need for automation, which calls for keeping as much structured information as possible.

The goal of this paper is to explore the design space for privacy-preserving audit with automation supported by formal logical representation of audit policies. In particular, we develop a system in which Hierarchical Identity Based Encryption (HIBE) [11] is used in coordination with a logic-based audit algorithm [15] to limit the amount of information the auditor needs to know to carry out access audit. The system uses HIBE to encrypt sensitive data on the audit logs with a ranking of sensitivity. The auditor uses the audit algorithm to decide which portions of the log need to be decrypted to provide evidence of the audit decision. We use HIBE because it provides a convenient way to encrypt audit

logs at a fine granularity (i.e., each event is encrypted using keys derived from the identifiers participating in that event) and limit the auditors to decrypt minimum-necessary data for audits. Since events occur frequently, but audits that require decryption of data are (presumed to be) rare, our framework provides an efficient audit procedure. We also explore the idea of extending the audit algorithm to provide understandable explanations for accesses rather than just claiming that an access can be proved consistent with or violate the policy. We design an audit architecture that augments an HIE using the ATNA [4] profile, a standard for auditing HIEs, and supports HIBE encryption of audit logs and an explanation-enabled audit procedure. A key feature of the design inspired by a pair of state HIEs (namely those for Maryland and Illinois) is the ability to combine the ATNA data with external documentation that is input to the audit algorithm. Our main contribution lies in the design and implementation of a practical system using this novel encryption and audit algorithm.

The rest of this paper is organized as follows. In Section 2, we provide background on HIE and HIBE. An overview of the architecture of our design is presented in Section 3. Next, we detail our hierarchical encryption algorithm (Section 4), the audit algorithm, and an example of using our infrastructure to audit accesses and produce explanations of the audit results (Section 5). We present our prototype implementation and evaluation results in Section 6. In the end, we discuss related work (Section 7) and conclude (Section 8).

2. BACKGROUND

Audit Standard for HIE. Because an HIE shares access to patient records to a broad community, it raises significant concerns about security and privacy [26]. To help address these concerns, the National Institute of Standards and Technology (NIST) recommends collecting and communicating audit logs about security-related events and data that should be gathered, consistent with applicable policy, as one of twelve services [23] to aid security and privacy protection for HIEs. The Audit Trail and Node Authentication (ATNA) Profile [4], is a standard developed by Integrating the Healthcare Environment (IHE) that provides an audit mechanism reflecting such security guidelines, with two strengths. First, it generates and keeps audit records in a centralized audit repository, which can be readily adapted to the typical HIE environment involving Cross Document Sharing (XDS) [4] and is compliant with HIPAA [3]. IHE stipulates that events defined in XDS.b [5] must be recorded in the audit trail. Second, the ATNA is compatible with diverse types of healthcare enterprises since it generates audit logs based on existing standards. In short, ATNA provides an infrastructure for auditing the HIE. However, it is still necessary to develop ways to stipulate policies and carry out the analysis of audit events on ATNA standard audit logs. Moreover, it may be necessary to work with external documentation that is not present in ATNA logs to achieve the overall goals of auditing the HIE. For instance, if one wishes to use billing records to confirm that an access to a record has been made to provide a (billed) service, then it may be desirable to integrate billing records into the underlying analytics engine rather than trying to make billing records a part of ATNA.

Hierarchical Identity Based Encryption. Hierarchical Identity Based Encryption (HIBE)[11, 24, 10] is a form of Identity Based Encryption (IBE) for hierarchical structures. IBE allows a sender to encrypt messages based on a receiver's identity, such as an e-mail address, before the receiver gets a secret decryption key from a Key Generation Center (KGC). The KGC generates a secret key to a user commensurate with the level of sensitivity of the data that the user needs to access. The private key will allow access at that level, or depth, in the hierarchy and at all lower levels and will also delegate authority to the holder of the private key to generate secret keys to users at lower levels. In other words, this is one-way access and delegation, so that a user at level k can generate a secret key for a user at any level lower than k, but lower level users can not use their secret keys to make secret keys for users at higher levels in the hierarchy. Upper level users who generate lower level keys are referred to as parents in the hierarchical structure. The HIBE system uses the following five algorithms.

$\mathsf{HIBE.Setup} : (k, \mathsf{L}) \mapsto \{mk, \mathsf{Pub}\}$ takes security parameter k and maximum depth L, and outputs a master key mk and public parameters Pub.

$\mathsf{HIBE.Extract} : (\mathsf{Pub}, mk, \mathsf{ID}) \mapsto sk_{\mathsf{ID}}$ takes the public parameters Pub, the master key mk and an identity $\mathsf{ID} = (id_1, ..., id_\ell)$ of depth $\ell (\leq \mathsf{L})$, and outputs a secret key sk_{ID} for the identity ID.

$\mathsf{HIBE.Delegate} : (\mathsf{Pub}, sk_{\mathsf{ID}'}, \mathsf{ID}) \mapsto sk_{\mathsf{ID}}$ takes the public parameters Pub, a parent's secret key $sk_{\mathsf{ID}'}$ where the parent's identity $\mathsf{ID}' = (id_1, ..., id_{\ell'})$ of depth $\ell' (< \ell \leq \mathsf{L})$, and the child's identity $\mathsf{ID} = (id_1, ..., id_\ell)$ of depth ℓ. Then the algorithm outputs a secret key sk_{ID} for the identity ID.

$\mathsf{HIBE.Encrypt}(\mathsf{Pub}, \mathsf{ID}, \mathsf{Msg}) \mapsto \mathsf{CT}$ takes the public parameters Pub, an identity ID and a message Msg, and outputs a ciphertext CT.

$\mathsf{HIBE.Decrypt} : (\mathsf{Pub}, sk_{\mathsf{ID}}, \mathsf{CT}) \mapsto \mathsf{Msg}$ takes the public parameters Pub, a secret key sk_{ID} and a ciphertext CT, and outputs a message Msg.

In Section 4, we describe how to use HIBE for a workable system to encrypt, transmit, and store external documentation for later use by HIE auditors.

3. AUDIT ARCHITECTURE

Figure 1 shows a high-level schema of the HIE and our audit subsystem. As data is accessed in the HIE on a day-to-day basis, a significant amount of internal ATNA-based audit information is gathered by the HIE. This information includes data such as which specific user accessed a patient's Protected Health Information (PHI) and which specific pieces of the PHI were accessed. Individual healthcare organizations (HCOs) using the HIE offer supplementary audit logs (external documentation) such as patient registration (PR) and medical billing (BL). The prescription monitoring program (PMP) [2], a centralized repository managed by the state of Illinois, provides prescription details. Together, the three justify HCOs' medical providers' accesses to the HIE. During the audit information collection by the audit data processor (ADP), the identifier cross-reference manager (ID-CRM) matches each external party's internal patient ID, user ID and hospital ID with the HIE's IDs. However, because the HCOs may have privacy concerns about such information, we allow them (or ADP) to encrypt the sensitive information using HIBE. The information is decrypted by the audit subsystem only when it is necessary for an audit. We describe details for doing this in

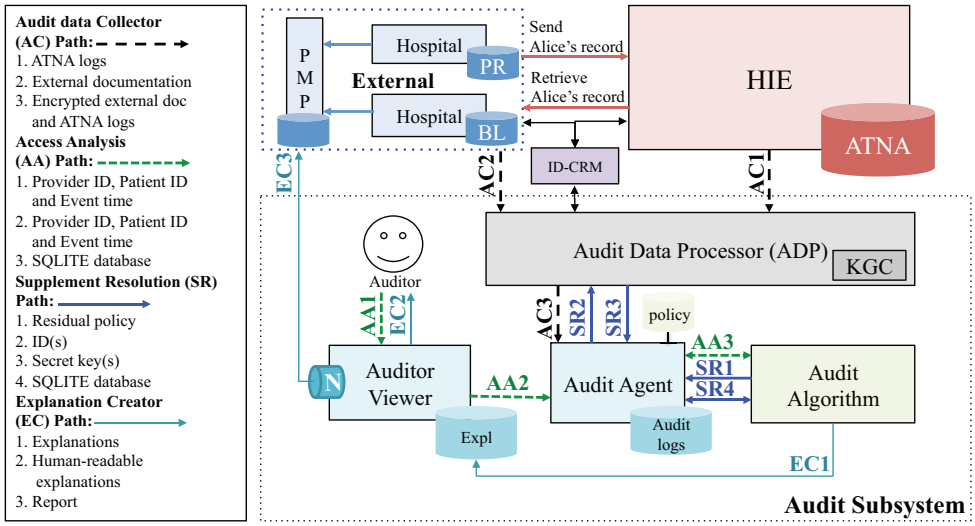

Figure 1: HIE Audit Infrastructure

Section 4. Our audit subsystem consists of four main components: (1) The ADP, which aggregates internal ATNA logs and external documentation from HCOs (called audit logs), encrypts external data and makes both available to the audit agent (path AC1 and AC2), (2) the audit agent, which is the main driver of remaining audit subsystem, (3) the audit algorithm, which is the core automation technology, and (4) an auditor viewer, which is the audit subsystem's interface for communicating with the auditor. The ADP contains the KGC for HIBE and must be trusted by the HCOs.

Access Analysis. The audit agent, which has the privacy policy relative to which audit is performed (represented in formal logic in our implementation), fetches relevant audit logs from the ADP (path AC3). The auditor sends a particular set of accesses of audit interest to the audit agent through the auditor viewer (paths AA1 and AA2). The audit agent passes the audit logs to the core audit algorithm in SQLite format (path AA3) along with the policy and the specific accesses of interest. The audit algorithm is designed to automatically check whether the specific access complies with the policy, if all relevant logs are available to it.

Supplement Resolution. Since some parts of the audit logs are encrypted, the audit algorithm may not be able to decide compliance or violation and return a residual policy (path SR1) which can be checked after the encrypted data has been decrypted (we describe the residual policy and its generation formally in Section 5).The audit agent determines what encrypted minimum information is needed to complete the audit by looking at the residual policy, and requests corresponding decryption keys from the ADP, more specifically the KGC (path SR2). Once the KGC returns the keys (path SR3), the audit agent decrypts the relevant external audit logs and sends the decrypted data to the audit algorithm as a SQLite database (path SR4). The ADP and the audit agent separately log the IDs whose keys were requested and the audit agent may additionally log the residual policies to facilitate a later audit on the audit process itself. The audit algorithm uses the newly available decrypted data to complete the audit.

Explanation Creator. The audit algorithm generates a formal explanation of why the access in question looks legitimate. This explanation is returned to the auditor viewer (path EC1), which then translates the explanation to human-readable form and returns it to the auditor (path EC2). If a policy violation is identified, the auditor viewer sends a notification with details of the audit log to the relevant HCOs (path EC3).

Our core audit algorithm (described in Section 5) is based on a prior algorithm called reduce, which can audit for policy violations even when some log information is missing [15]. Over this basic algorithm, we make two additional contributions. First, we extend reduce's output with explanations showing why a certain access satisfies or violates the policy. Second, we allow HCOs or the ADP to encrypt external audit logs in layers, through HIBE. This allows the audit agent to selectively decrypt minimum-necessary data, and facilitates any subsequent audit of the audit process and might increase the HCOs' confidence in the HIE. Use of HIBE (instead of conventional public-key cryptography) also allows the HCOs to encrypt data offline, without any interaction with the KGC.

4. HIERARCHICAL ENCRYPTION

In this section, we describe how we encrypt and decrypt external audit logs, specifically documentation, from the HCOs using HIBE for privacy protection during audits. There are various HIBE schemes that satisfy different properties and security goals. It is important to select a HIBE scheme with constant, small-size ciphertexts, e.g. [11], since such a scheme will optimize storage cost regardless of hierarchy depth. Even though privacy infringement investigations are rare in practice and encrypted sensitive data will seldom be decrypted, it must nonetheless be stored by the audit subsystem.

External Data Hierarchy and Identity. Suppose that the data in some external documentation $\mathcal{D}$ has been partitioned hierarchically into n degrees of sensitivity, resulting in data $\mathcal{D}_1, \ldots, \mathcal{D}_n$, where $\mathcal{D}_n$ is the data in the documentation

patient-id	HCO-id	date-of-bill	level1	level2	level3
eeb728473e1949a	Carle07RQ12	2013:09:08:10:18:41	$Enc_{\mathsf{ID}_{1,1}}$(cs1010)	$Enc_{\mathsf{ID}_{1,2}}$(HEMOGLO...)	$Enc_{\mathsf{ID}_{1,3}}$(73.8)
d99486a44ca64cb	Provena01AV98	2013:09:17:02:48:29	$Enc_{\mathsf{ID}_{2,1}}$(ra1010)	$Enc_{\mathsf{ID}_{2,2}}$(MCH,Auto...)	$Enc_{\mathsf{ID}_{2,3}}$(279)
42210b2417d74b1	NWM0329W2	2013:10:21:11:47:22	$Enc_{\mathsf{ID}_{3,1}}$(pq1010)	$Enc_{\mathsf{ID}_{3,2}}$(PLATELET...)	$Enc_{\mathsf{ID}_{3,3}}$(11.6)

Table 1: observes-in-bill table

that is most sensitive, and $\mathcal{D}_1$ is the data that is the least sensitive. In our system, all data in $\mathcal{D}_n$ will be encrypted using an IBE identifier $\mathsf{ID}_n = (id_1)$. We suggest this identifier be derived from non-sensitive descriptive information such as the HCO's identity, the patient's identity, degree of sensitivity, and time. Data in $\mathcal{D}_{n-1}$ will be assigned an identifier $\mathsf{ID}_{n-1} = (id_1, id_2)$ and so on. Finally, data in $\mathcal{D}_1$ will be assigned an identifier $\mathsf{ID}_1 = (id_1, ..., id_n)$. As an example, the billing table shown in Table 1 is hierarchically organized into 3 sensitivity levels. Information concerning observation result that a patient receives is level 3 sensitive (most sensitive), observation type is level 2 sensitive, and provider information is level 1 sensitive (least sensitive).

Billing Data Encryption. Table 1 shows encrypted patient observation information taken from illustrative medical bills. Consider data $\mathcal{D}$ that contains exactly the first row of the table. Within $\mathcal{D}$, data at the highest level of sensitivity, $\mathcal{D}_3(=$ 73.8), will be assigned identity $\mathsf{ID}_3 = (id_1)$ which is the concatenation eeb728473e1949a || Carle07RQ12 || 2013:09:08:10:18:41 || level3 of non-sensitive identifying information. The identity of less sensitive data, $\mathcal{D}_2(=$ HEMOGLO...), is $\mathsf{ID}_2 = (id_1, id_2)$, where $id_2 =$ level2. Finally, the identity of the least sensitive data $\mathcal{D}_1(=$ cs1010) is $\mathsf{ID}_1 = (id_1, id_2, id_3)$, where $id_3 =$ level1. External documentation at each level of sensitivity $(\mathcal{D}_1, ..., \mathcal{D}_n)$ is encrypted using the assigned identities ID_i $(1 \leq i \leq n)$. For $i = 1$ to n, the ADP or the HCOs run $\mathsf{HIBE.Encrypt}(\mathsf{Pub}, \mathsf{ID}_i, \mathcal{D}_i)$ to get a corresponding ciphertext $Enc_{\mathsf{ID}_i}(\mathcal{D}_i)$. For example, $Enc_{\mathsf{ID}_{1,3}}$(73.8) in Table 1 represents the encryption of the data $\mathcal{D}_3$. The corresponding HIBE encryption would be $\mathsf{HIBE.Encrypt}(\mathsf{Pub}, \mathsf{ID}_{1,3}, 73.8)$ where $\mathsf{ID}_{1,3} = \mathsf{ID}_3$.

This example is illustrative: All external documentation in our system is organized in tables and encrypted with HIBE. Identities used for encryption may have cell-, column- or table-granularity, depending on the nature of the documentation.

Issuance of Secret Keys and Decryption. When the audit algorithm requests encrypted data, the audit agent requests an appropriate secret key from the KGC for decryption. Assume that the audit agent needs the secret key corresponding to ID_k. To issue an appropriate secret key to the audit agent, the KGC runs $\mathsf{HIBE.Extract}(\mathsf{Pub}, mk, \mathsf{ID}_k)$ using its master key mk to get sk_{ID_k}. After receiving sk_{ID_k} from the KGC, the audit agent runs $\mathsf{HIBE.Decrypt}(\mathsf{Pub}, sk_{\mathsf{ID}_k}, Enc_{\mathsf{ID}_k}(\mathcal{D}_k))$ to obtain $\mathcal{D}_k$. If the audit agent needs additional information at any level k' that is lower than k, the audit agent runs $\mathsf{HIBE.Delegate}(\mathsf{Pub}, sk_{\mathsf{ID}_k}, \mathsf{ID}_{k'})$ to get a secret key $sk_{\mathsf{ID}_{k'}}$, and then runs $\mathsf{HIBE.Decrypt}(\mathsf{Pub}, sk_{\mathsf{ID}_{k'}}, Enc_{\mathsf{ID}_{k'}}(\mathcal{D}_{k'}))$ to get $\mathcal{D}_{k'}$. This does not require communication with the KGC.

5. AUDIT WITH EXPLANATIONS

Garg et al. [15] develop an algorithm for finding violations of a policy on system logs. Their algorithm takes into consideration incompleteness of information in logs. For example, if the policy carries the obligation "a notice must be sent in the next 30 days", then before the 30 day deadline is reached, the log may not contain enough information to decide whether or not this obligation is met. To account for incompleteness, the reduce algorithm uses a best-effort (but sound) approach; it checks as much of the policy as possible given the available log and returns a residual policy that captures policy conditions which could not be verified. When the missing information becomes available, the residual policy can be re-checked using the reduce algorithm itself. Here, we use reduce's support for incompleteness to audit iteratively; as iteration rounds progress, the audit algorithm decrypts increasingly sensitive, HIBE-encrypted external audit logs. This continues until either it is determined that there are no policy violations or relevant violations have been found. Thus, we limit the amount of external audit logs decrypted for audit.

It also helps audit to have an intuitive explanation of why a decision (policy violation or not) was made. For instance, it is helpful to know that a physician's access to a medical record in a hospital was allowed because the patient was referred to that hospital, or because that the patient visited that facility. Accordingly, we extend the reduce algorithm to also provide such an explanation.

5.1 Policy and Explanation Syntax

Following the prior work [15], we use a first-order logic as the policy specification language. We summarize the syntax of formulas and explanations in Figure 2. We write α to denote formulas and φ to denote generalized formulas, which are either formulas or audit decisions ($\top$ = no violation, $\bot$ = violation) coupled with explanations γ. Formulas include atomic predicates, true ($\top$), false ($\bot$), conjunctions ($\wedge$) and disjunctions ($\vee$) of formulas (denoted C and D, respectively), and first-order quantifiers ($\forall$ and $\exists$).

Here c denotes a restricted class of formulas, called guards, borrowed from [15]. They guarantee (statically) that the number of substitutions for x that makes c true is always finite. When guarded by a formula c, both universal and existential quantifiers can be handled easily. Readers may ignore the distinction between c and α for the purpose of understanding this section.

Each formula is annotated with a policy label, written ℓ. Labels have no semantic meaning except to establish a syntactic link between an explanation and the original formula from which the explanation was derived. The formula $\sigma \rhd \varphi$ means that the substitution for free variables in φ is σ. This formula itself is not used for policy specification. It can appear in residual formulas output by our extended reduce algorithm.

An explanation γ corresponds to a sub-tree of labels of a formula's abstract syntax tree. An explanation is only meaningful relative to a formula. An explanation can be a single label, which points to a leaf position of the formula.

Conj clause	C	::=	$\bigwedge_i \varphi_i$
Disj clause	D	::=	$\bigvee_i \varphi_i$
Formula	α	::=	$\langle\ell\rangle P \mid \langle\ell\rangle\top \mid \langle\ell\rangle\bot \mid \langle\ell\rangle C \mid \langle\ell\rangle D$
		$\mid$	$\langle\ell\rangle\forall\vec{x}.(c \supset \varphi) \mid \langle\ell\rangle\exists\vec{x}.(c \wedge \varphi)$
		$\mid$	$\sigma \rhd \varphi$
Generalized form.	φ	::=	$\alpha \mid \texttt{expl}(\top,\gamma) \mid \texttt{expl}(\bot,\gamma)$
Explanation	γ	::=	$\ell \mid \ell \circ \gamma \mid \gamma_1 \oplus \gamma_2 \mid \sigma \rhd \gamma$

Figure 2: Syntax of formulas and explanations

A concatenated explanation $\ell \circ \gamma$ is an explanation for a formula labeled by ℓ at the root, where γ is, recursively, the explanation of the root's children. An explanation can also combine explanations from branches of a conjunction or disjunction (denoted $\gamma_1 \oplus \gamma_2$). Finally, an explanation can be guarded by a substitution σ (syntax: $\sigma \rhd \gamma$).

5.2 Extended Reduce Algorithm

The reduce algorithm, as presented in [15], takes as argument a policy formula α and an audit log $\mathcal{L}$ and returns a residual policy α', which may be $\top$ (no violation), $\bot$ (violation) or another formula called the residual formula (meaning that critical information is absent from the log; α' must be checked when more information is available). To extend reduce to generate explanations, we change it to take as input a policy represented as a generalized formula φ, a log $\mathcal{L}$ and, additionally, a substitution σ for free variables of φ. The output of our extended reduce algorithm is also a generalized formula. If the output is α, it means that some information necessary for audit is missing from the log, and α is the residual policy to be checked when that information becomes available. The output $\texttt{expl}(\top,\gamma)$ means that there is no policy violation and γ explains why that is the case. Similarly, the output $\texttt{expl}(\bot,\gamma)$ signals a policy violation justified by explanation γ.

We define a function simplify that takes a generalized formula and returns another generalized formula, in simpler form. The function simplify serves a dual purpose. First, it rewrites the original formula using basic rules of logic, e.g., it replaces $\varphi \wedge \top$ with φ. Second, and more importantly, if the input formula is equivalent to either $\top$ or $\bot$, it produces a succinct explanation of why that is the case by combining and selectively retaining explanations from the original formula. The output of simplify is either a residual policy formula α, or a binary answer ($\top$ or $\bot$) paired with an explanation γ.

We omit the detailed definitions of reduce and simplify; next, we demonstrate how the algorithm works.

5.3 Example Scenario

In this section, we present a simple audit scenario that illustrates reduce, simplify and HIBE.

Policy. Suppose a HIE's policy for data sharing is that a provider p_1 can send detailed information about a patient q to another provider p_2 if, within a year of such sharing, p_2 bills the insurance company for services provided to q at p_2. The encoding of a HIE's policy for data sharing is shown below. A provider p_1 can send a patient document m to a provider p_2 at time t ($\mathsf{send}(p_1,p_2,m,t)$), where (1) m describes patient q ($\mathsf{hasattrof}(m,q)$), (2) m includes detailed information ty and va about the patient, e.g., ty is the type of observation q is under and va is the result of the observation ($\mathsf{includes}(m,ty,va,t)$), (3) q is classified as type tp and provided with service vl at t ($\mathsf{patientInfo}(q,tp,vl,t)$), (4) p_2 works in organization o ($\mathsf{organization}(p_2,o,t)$), and (5) organization o records that patient q has an insurance plan p from company c at time t ($\mathsf{insuranceInfo}(q,p,c,t)$); then there should be a consequent patient medical bill of type b at time t' ($\mathsf{medical\text{-}bill}(q,b,t')$), t' should be within 365 days of the data sharing, the organization o should note that q has an insurance plan c with company p ($\mathsf{insurance}(q,p,c,o,t')$), and either the bill is from q's visit to p_2 or for an observation carried out by p_2 on q. Predicates $\mathsf{visits\text{-}in\text{-}bill}(q,p_2,vl,o,t')$ and $\mathsf{observes\text{-}in\text{-}bill}(q,p_2,ty,va,o,t')$ represent p_2's records of medical bills of the two specific types.

$$
\begin{aligned}
&\varphi_{pol} = \langle\texttt{DISC}\rangle\\
&\forall p_1,p_2,m,q,t,ty,va,tp,vl,o,p,c\\
&\quad \mathsf{send}(p_1,p_2,m,t)\wedge\ \mathsf{hasattrof}(m,q)\wedge\\
&\quad \mathsf{includes}(m,ty,va,t)\wedge\ \mathsf{patientInfo}(q,tp,vl,t)\wedge\\
&\quad \mathsf{organization}(p_2,o,t)\wedge\ \mathsf{insuranceInfo}(q,p,c,t)\\
&\quad \supset\langle\texttt{AC}\rangle\exists t',b.\mathsf{medical\text{-}bill}(q,b,t')\wedge\\
&\qquad \langle\texttt{BLL}\rangle(\langle\texttt{time}\rangle\mathsf{timein}(t,t',t+365)\\
&\qquad\quad \wedge\langle\texttt{INS}\rangle\mathsf{insurance}(q,p,c,o,t')\\
&\qquad\quad \wedge\langle\texttt{DJ}\rangle(\langle\texttt{VST}\rangle(\langle\texttt{B}\rangle b = \textit{visit-history}\wedge\\
&\qquad\qquad\qquad \langle\texttt{visit}\rangle\mathsf{visits\text{-}in\text{-}bill}(q,p_2,vl,o,t')))\\
&\qquad\qquad \vee(\langle\texttt{OBS}\rangle(\langle\texttt{B}\rangle b = \textit{observation}\wedge\\
&\qquad\qquad\qquad \langle\texttt{obsv}\rangle\mathsf{observes\text{-}in\text{-}bill}(q,p_2,ty,\\
&\qquad\qquad\qquad\qquad va,o,t'))))
\end{aligned}
$$

Audit Logs. The internal log contains information about all the predicates to the left of the implication in φ_{pol} as well as the predicate medical-bill. Predicates visits-in-bill and observes-in-bill record detailed information about patients' hospital visits, which are considered external documentations that belong to the hospital. Both external and internal logs are represented as database tables. Tables visits-in-bill and observes-in-bill are HIBE-encrypted according to levels shown in Table 2.

For illustration, we assume that the following substitution σ is the only one that satisfies the condition of the outermost universal quantification on the example log (here, terms like $P1$ starting with uppercase letters are constants):

$$
\begin{aligned}
\sigma = {} & p_1{\mapsto}P1, p_2{\mapsto}P2, m{\mapsto}M1, q{\mapsto}Q1, t{\mapsto}T1, ty{\mapsto}TY1,\\
& va{\mapsto}VA1, tp{\mapsto}TP1, vl{\mapsto}VL1, o{\mapsto}O1, p{\mapsto}PI, c{\mapsto}C1
\end{aligned}
$$

We further assume that $\mathsf{timein}(T1, T2, T1 + 365)$ and $\mathsf{timein}(T1, T3, T1 + 365)$ are true. Below are the (only) log entries about predicates to the right of $\supset$ in φ_{pol}.

$\mathsf{medical\text{-}bill}(Q1, \textit{visit-history}, O1, T2)$
$\mathsf{medical\text{-}bill}(Q1, \textit{observation}, O1, T3)$
$\mathsf{visits\text{-}in\text{-}bill}(Q1, P2, VL1, O1, T2)$
$\mathsf{observes\text{-}in\text{-}bill}(Q1, P2, TY2, VA2, O2, T3)$
$\mathsf{insurance}(Q1, PI, C1, O1, T2)$

Reduce on Encrypted Data. In the initial phase of audit, the auditor does not possess decryption keys for external data. This poses no problem because reduce can handle log incompleteness; reduce treats the log incomplete in predicates like visits-in-bill, and simply returns such predicates in the residual output. The output of running reduce on φ_{pol} and the example log is shown in the next page.

Level 1 (least)	Level 2	Level 3 (most)
provider-id (p_2)	service-type (vl)	OBS-value(va)
INS-company (p)	INS-plan (c)	
	OBS-type (ty)	

Table 2: Sensitivity levels in the audit scenario

$$\varphi_{r1} = \langle \texttt{DISC} \rangle \sigma \rhd \\ \langle \texttt{AC} \rangle \sigma_1 \rhd \langle \texttt{BLL} \rangle \\ (\langle \texttt{time} \rangle \top \wedge \langle \texttt{INS} \rangle \mathsf{insurance}(q,p,c,o,t') \wedge \\ \langle \texttt{DJ} \rangle (\langle \texttt{VST} \rangle (\langle \texttt{B} \rangle \top \wedge \langle \texttt{visit} \rangle \mathsf{visits\text{-}in\text{-}bill}(q,p_2,vl,o,t')) \\ \vee \langle \texttt{OBS} \rangle (\langle \texttt{B} \rangle \bot \wedge \\ \langle \mathsf{obsv} \rangle \mathsf{observes\text{-}in\text{-}bill}(q,p_2,ty,va,o,t'))) \\ \vee \sigma_2 \rhd \langle \texttt{BLL} \rangle \\ (\langle \texttt{time} \rangle \top \wedge \langle \texttt{INS} \rangle \mathsf{insurance}(q,p,c,o,t') \wedge \\ \langle \texttt{DJ} \rangle (\langle \texttt{VST} \rangle (\langle \texttt{B} \rangle \bot \wedge \langle \texttt{visit} \rangle \mathsf{visits\text{-}in\text{-}bill}(q,p_2,vl,o,t')) \\ \vee \langle \texttt{OBS} \rangle (\langle \texttt{B} \rangle \top \wedge \\ \langle \mathsf{obsv} \rangle \mathsf{observes\text{-}in\text{-}bill}(q,p_2,ty,va,o,t')))$$

Here, $\sigma_1 = t' \mapsto T2, b \mapsto \textit{visit-history}$
$\sigma_2 = t' \mapsto T3, b \mapsto \textit{observation}$

The existentially quantified variables t' and b in φ_{pol} have two possible substitutions, corresponding to the two entries in the medical-bill table. The residual formula hense contains a disjunction over these two possibilities.

Simplification. Next, we call simplify on φ_{r1} to condense as many explanations as possible. This yields:

$$\varphi_{s1} = \langle \texttt{DISC} \rangle \sigma \rhd \\ \langle \texttt{AC} \rangle \sigma_1 \rhd \langle \texttt{BLL} \rangle \\ (\texttt{expl}(\top, \texttt{time}) \wedge \langle \texttt{INS} \rangle \mathsf{insurance}(q,p,c,o,t') \wedge \\ \langle \texttt{DJ} \rangle (\langle \texttt{VST} \rangle (\texttt{expl}(\top, \texttt{B}) \wedge \\ \langle \texttt{visit} \rangle \mathsf{visits\text{-}in\text{-}bill}(q,p_2,vl,o,t') \\ \vee \texttt{expl}(\bot, \texttt{OBS} \circ \texttt{B}))) \\ \vee \sigma_2 \rhd \langle \texttt{BLL} \rangle \\ (\texttt{expl}(\bot, \texttt{time}) \wedge \langle \texttt{INS} \rangle \mathsf{insurance}(q,p,c,o,t') \wedge \\ \langle \texttt{DJ} \rangle (\texttt{expl}(\bot, \texttt{VST} \circ \texttt{B}) \\ \vee \langle \texttt{OBS} \rangle (\texttt{expl}(\top, \texttt{B}) \wedge \\ \langle \mathsf{obsv} \rangle \mathsf{observes\text{-}in\text{-}bill}(q,p_2,ty,va,o,t'))))$$

Requesting Decryption Keys. To proceed further, we must decrypt the insurance table and either the observes-in-bill table or the visits-in-bill table. Since the maximum sensitivity level of entries in visits-in-bill is lower than that in observes-in-bill, the audit agent asks the KGC for keys at level 2 (the maximum level of visits-in-bill and insurance).[1] The KGC generates keys based on its master key mk and gives them to the audit agent. Both the KGC and the audit agent may log why the keys were generated (by recording the residual formula φ_{s1}) to aid a subsequent audit of this audit process. The audit agent then decrypts entries in those two tables and provides the formula φ_{s1} with the decrypted tables (added to the original log) to reduce. The output of reduce is the following formula φ_{d1}. Note that the clause guarded by σ_2 remains the same because even the extended log contains no information to reduce it.

$$\varphi_{d1} = \\ \langle \texttt{DISC} \rangle \sigma \rhd \\ \langle \texttt{AC} \rangle \sigma_1 \rhd \langle \texttt{BLL} \rangle \\ (\texttt{expl}(\top, \texttt{time}) \wedge \texttt{expl}(\top, \texttt{INS}) \wedge \\ \langle \texttt{DJ} \rangle (\langle \texttt{VST} \rangle (\texttt{expl}(\top, \texttt{B}) \wedge \texttt{expl}(\top, \texttt{visit})) \vee \texttt{expl}(\bot, \texttt{OBS} \circ \texttt{B}))) \\ \vee \sigma_2 \rhd \langle \texttt{BLL} \rangle \\ (\texttt{expl}(\bot, \texttt{time}) \wedge \langle \texttt{INS} \rangle \mathsf{insurance}(q,p,c,o,t') \wedge \\ \langle \texttt{DJ} \rangle (\texttt{expl}(\bot, \texttt{VST} \circ \texttt{B}) \\ \vee \langle \texttt{OBS} \rangle (\texttt{expl}(\top, \texttt{B}) \wedge \\ \langle \mathsf{obsv} \rangle \mathsf{observes\text{-}in\text{-}bill}(q,p_2,ty,va,o,t'))))$$

[1]For simplicity, we assume here that the audit agents decrypts entire tables atomically. In practice, it could decrypt only specific rows of interest.

Simplification and Explanation. Finally, simplify is run on φ_{d1} to obtain the following output:

$$\varphi_{s2} = \texttt{expl}(\top, \texttt{DISC} \circ \sigma \rhd \texttt{AC} \circ \sigma_1 \rhd \langle \texttt{BLL} \rangle \circ \\ (\texttt{time} \oplus \texttt{INS} \oplus (\texttt{DJ} \circ \texttt{VST} \circ (\texttt{B} \oplus \texttt{visit})))$$

The result indicates that the log satisfies the policy. The reason is the following: (1) σ is the only substitution that makes the conditions associated with the action send true and (2) the conditions required for such a send (AC) under σ are true. The explanation of (2) is that there exists a substitution σ_1 that matches an entry in medical-bill and makes BLL true. More concretely, the time of the bill, the insurance information, and hospital's billing record of the patient all satisfy the policy constraints. In particular, the hospital's record shows that the patient visited the hospital (VST).

6. IMPLEMENTATION & EVALUATION

To validate our proposal, we implement the HIE (Figure 1) based on IHE Profile XDS.b[5]. It supports the sharing of patient clinical documents based on the HIE's document registry which keeps a patient document index and location where the documents are stored in. Based on these, the web-based document viewer looks for patient documents based on patient information. We implement a Java API to create ATNA-based XML logs [19] on top of HIE. We report our evaluation of HIBE key generation, decryption and the reduce algorithm based on a policy encoding the guidelines [6] of the U.S. Office of the National Coordinator (ONC) for Health Information Technology and a synthetic audit log. All experiments are performed on a machine with an Intel Core i7 2.3GHz processor and 1GB of memory, running Ubuntu 12.04. We use the Charm library [1] to implement the HIBE module. In particular, we use a symmetric curve with a 512-bit base field to initiate a group in the elliptic curve with bilinear pairings. For illustration purposes, we assume a maximum depth of three sensitivity levels in the hierarchy. To encrypt arbitrary messages with HIBE, we use a hybrid encryption scheme: we extract a session key after hashing [10] a random element from the message space of HIBE, encrypt messages with the session key via AES (CBC mode) symmetric encryption and encrypt the random element using HIBE [11].

Policy. According to the ONC, providers requesting a patient's IIHI (individually identifiable health information) by electronic means for treatment must verify a treatment relationship with a patient by attestation or artifacts such as patient registration, prescriptions, consults, and referrals. The top-level encoding of the policy is shown below and consists of a disjunction of six sub clauses, and at least one must be satisfied for each access. We omit the details of these clauses. φ_{pol}, shown in Section 5.3, is a simplified encoding of $\varphi_{Billing}$.

$$\varphi_{ONC} = \forall p_1, p_2, m, q, t, ty, va, tp, vl, o, p, c \\ \mathsf{send}(p_1, p_2, m, t) \wedge \mathsf{hasattrof}(m, q) \wedge \\ \mathsf{includes}(m, ty, va, t) \wedge \mathsf{patientInfo}(q, tp, vl, t) \wedge \\ \mathsf{organization}(p_2, o, t) \wedge \mathsf{insuranceInfo}(q, p, c, t) \\ \supset \varphi_{Exception} \vee \varphi_{Billing} \vee \varphi_{Registration} \vee \varphi_{Prescription} \\ \vee \varphi_{Referral} \vee \varphi_{Consult}$$

Audit Logs. We generate synthetic data representing both external audit logs and internal ATNA logs. The generated ATNA log has a size of 5.7 MB, which represents

	Key Gen. (ms)	Session key Dec.(ms)	Message Dec.(ms)	Algorithm (ms)	Single access(ms)	Day (s)	Month (m)
`Up to Level 3`	17.73	20.76	0.06	42.6	81.15	6.57	3.26
`Up to Level 2`	11.73	13.73	0.04	41.3	66.80	5.41	2.68

Table 3: Consumption time for HIBE and reduce

9,644 accesses to HIE over 4 months (this realistic number is based on Johnson *et al.*'s data [18]). The external audit data has a size of 12 MB, and includes roughly 9,644 entries about patient registration, billing and referral. The external logs are encrypted using HIBE with three pre-defined sensitivity levels.

Evaluation Results. We evaluate the efficiency and scalability of both HIBE and the reduce algorithm. We use the audit scenario shown in Section 5.3 and break it into three phases. In the first phase, the auditor does not have any keys, and we measure the time reduce takes to generate a residual policy; in the second phase, we measure the time it takes the KGC to generate a decryption key given an ID, and the time it takes to decrypt relevant log data using the key (because our encryption is hybrid, the latter further splits into the time taken to decrypt the symmetric key, and the time take to decrypt the data using the symmetric key); in the third phase, we run reduce again and measure the time reduce takes to check the residual policy on the decrypted data. We run the audit scenario on two accesses, one requires a decryption key of level 2, and the other requires a decryption key of level 3. The size of messages encrypted up to level 2, shown in Table 3, is 416B including the session key and the size of messages encrypted up to level 3 is 580B including the session key. Table 3 summarizes our results. All numbers are averages of 20 trials (all have negligible standard deviations). The first column shows the time taken to generate HIBE decryption keys, the second column indicates the time needed to decrypt a session key with the HIBE key, the third column shows the time to decrypt a message using AES and the fourth column shows the total time consumed by reduce, which is derived by adding up the time for each iteration of reduce(before and after decryption). As can be seen, the total time is split almost evenly between reduce and the cryptographic operations for data at level 3 and is dominated by reduce for data at level 2. Johnson et al [18] report approximately 81 accesses per day in a typical HIE. Based on this number, we calculate the total time for auditing all accesses in a day and in a month to be 6.57/5.41 seconds (level 3/level 2) and 3.26/2.68 minutes, respectively. Since audit is an offline process, we consider these numbers practical.

Practical Issues in Deployment. Integrating our audit architecture into an actual distributed healthcare environment will require the HCOs to trust the security of the audit infrastructure. Our audit subsystem mitigates this concern to some extent, since data is stored encrypted and it is decrypted only when absolutely necessary and auditors never see log data unencrypted. However, in cases where human judgment is necessary, auditors may need access to unencrypted data. This raises concerns about auditor trustworthiness, but such concerns are orthogonal to our design and exist in all audit environments.

7. RELATED WORK

Audits in the Healthcare Domain. Previous works have envisaged a scientific and technical approach to audit the healthcare system. Gunter et al. [17] introduce access rules informed by probabilities (ARIP) to establish appropriate access rules for HCOs based on their work flows and social networks by analyzing audit logs and attributes of HCOs. However, their work is not feasible for infrastructure currently in place at HCOs, while our work is applicable to the real-world heterogeneous healthcare environment with IHE standard based audit infrastructure. In addition, Fabbri and LeFevre [13, 14] have proposed explanation-based auditing, which enables patients to review access to their health records with human interpretable explanations. They adopt a machine learning approach to automatically generate log explanation while we use the logic-based algorithm to identify the legitimacy of access based on the privacy policy. Gregg [16] builds an audit interface for the ATNA-based audit logs from a picture archiving and communications system (PACS). However, their work does not consider privacy.

Audit Log Encryption. We share the goal of previous encryptions of audit logs[12, 22, 27] to not only protect against malicious attackers, but also to limit exposure of private information in the log to an authorized auditor. Our approach is different from previous methods for encrypting audit logs because we supplement internal log information with external data, which we encrypt with HIBE, allowing auditors to access only the minimum necessary data. In doing so, our scheme creates an identity, a descriptive label, using terms that do not need to be encrypted, such as unidentifiable codes for patient, provider, and type and date of visit, for the audit log entry, allowing the auditor to find the log needed without the more cumbersome encrypted keyword approach previously used. In previous schemes, such as [27], the auditor has to match all encrypted keywords with a given trapdoor that contains those words. This makes it secure but extremely inefficient.

Algorithms for Policy Compliance Checking. We build on Garg et al.'s algorithm reduce for auditing policies over incomplete logs [15]. We inherit the policy language used by the reduce algorithm, which is a first-order logic that can encode first-order Linear Temporal Logic (LTL) formulas. Policies that are naturally specified in LTL can be easily translated to formulas in this logic. There has been much work on compliance checking of policies expressed in Linear Temporal Logic (LTL) [25, 7, 20, 9, 8, 21]. Most of the work focuses on runtime monitoring. In contrast, we assume that logs are recorded by audit agents, and post-hoc audit is applied to these logs. To our knowledge, reduce is the only policy-based log audit algorithm that can handle incomplete logs. We leverage reduce's capability to handle incomplete logs to integrate encrypted logs. We also extend reduce to generate explanations for the output of the algorithm.

8. CONCLUSIONS

We have proposed an audit infrastructure for broker-based HIE systems, that limits the information shared through HIE. The audit logs are encrypted with HIBE and stored in a centralized audit repository for effective HIE audit, and decrypted on a need-only basis by our audit subsystem. Our logic-based audit algorithm provides further evidence of the auditor's behavior, and thus increases the trustworthiness of the system. The initial performance evaluation of a prototype implementation shows that our proposed infrastructure is practical and scalable. As future work, we plan to implement the extended audit algorithm with explanations and investigate the possibility of combining audit and access control mechanisms in the HIEs.

Acknowledgements

This work was partially supported by NSF CNS 09-64392 (NSF EBAM) and HHS 90TR0003-01 (SHARPS). The views expressed are those of the authors only. We appreciated feedback on this work from Mark Chudzinski and Ivan Handler from the Office of Health Information Technology of the State of Illinois and from Mike Berry and Noam Arzt of HLN Consulting, LLC.

9. REFERENCES

[1] Charm: A tool for rapid cryptographic prototyping. http://www.charm-crypto.com.

[2] Illinois Prescription Monitoring Program. https://www.ilpmp.org/.

[3] HIPAA Security Series, 4 Security Standards: Technical Safeguards. Department of Health and Human Services USA, 2007.

[4] Integrating the healthcare enterprise volume 1 integration profiles. ACC, HIMSS and RSNA Integrating the Healthcare Enterprise, 2007.

[5] IHE IT Infrastructure Technical Framework Supplement 2007-2008 Cross-Enterprise Document Sharing-b (XDS.b). ACC, HIMSS and RSNA Integrating the Healthcare Enterprise, 2008.

[6] Privacy and Security Framework Requirements and Guidance for the State Health Information Exchange Cooperative Agreement Program. Office of the National Coordinator for Health Information Technology, 2012.

[7] F. Baader, A. Bauer, and M. Lippmann. Runtime verification using a temporal description logic. In *Proc. of FroCos*, 2009.

[8] H. Barringer, A. Goldberg, K. Havelund, and K. Sen. Rule-based runtime verification. In *Proc. of VMCAI*, 2004.

[9] D. A. Basin, F. Klaedtke, and S. Müller. Policy monitoring in first-order temporal logic. In *Proc. of CAV*, 2010.

[10] D. Boneh and X. Boyen. Efficient selective identity-based encryption without random oracles. *Journal of Cryptology*, 24(4):659–693, 2011.

[11] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology–EUROCRYPT 2005*, pages 440–456. Springer, 2005.

[12] D. Davis, F. Monrose, and M. K. Reiter. Time-scoped searching of encrypted audit logs. In *Information and Communications Security*, pages 532–545. Springer, 2004.

[13] D. Fabbri and K. LeFevre. Explanation-based auditing. *Proc. VLDB Endowment*, 5(1), 2011.

[14] D. Fabbri and K. LeFevre. Explaining accesses to electronic medical records using diagnosis information. *Journal of the American Medical Informatics Association*, 20(1), 2013.

[15] D. Garg, L. Jia, and A. Datta. Policy auditing over incomplete logs: theory, implementation and applications. In *Proc. of CCS*, 2011.

[16] B. Gregg, H. D'Agostino, and E. Toledo. Creating an IHE ATNA-based audit repository. *Journal of Digital Imaging*, 2006.

[17] C. Gunter, D. Liebovitz, and B. Malin. Experience-based access management: A life-cycle framework for identity and access management systems. *Proc. of IEEE Security & Privacy*, 9(5), 2011.

[18] K. B. Johnson, K. M. Unertl, Q. Chen, N. M. Lorenzi, H. Nian, J. Bailey, and M. Frisse. Health information exchange usage in emergency departments and clinics: the who, what, and why. *Journal of the American Medical Informatics Association*, 18(5):690–7, 2011.

[19] G. Marshall. Security Audit and Access Accountability Message XML Data Definitions for Healthcare Applications. (September), 2004.

[20] G. Roşu and K. Havelund. Rewriting-based techniques for runtime verification. *Automated Software Engineering*, 12:151–197, 2005.

[21] M. Roger and J. Goubault-Larrecq. Log auditing through model-checking. In *Proc. of CSF*, 2001.

[22] B. Schneier and J. Kelsey. Secure audit logs to support computer forensics. *ACM Transactions on Information and System Security (TISSEC)*, 2(2):159–176, 1999.

[23] M. Scholl, K. Stine, K. Lin, and D. Steinberg. *Security Architecture Design Process for Health Information Exchanges (HIEs)*. 2010.

[24] J. H. Seo, T. Kobayashi, M. Ohkubo, and K. Suzuki. Anonymous hierarchical identity-based encryption with constant size ciphertexts. In *Public Key Cryptography–PKC 2009*, pages 215–234. Springer, 2009.

[25] P. Thati and G. Roşu. Monitoring algorithms for metric temporal logic specifications. *Electronic Notes in Theoretical Computer Science*, 113:145–162, 2005.

[26] J. R. Vest and L. D. Gamm. Health information exchange: persistent challenges and new strategies. *Journal of the American Medical Informatics Association*, 17(3):288–94, Jan. 2010.

[27] B. R. Waters, D. Balfanz, G. Durfee, and D. K. Smetters. Building an encrypted and searchable audit log. In *Proc. of NDSS*, 2004.