Understanding and Protecting Privacy: Formal Semantics and Principled Audit Mechanisms*

Anupam Datta[1], Jeremiah Blocki[1], Nicolas Christin[1], Henry DeYoung[1], Deepak Garg[2], Limin Jia[1], Dilsun Kaynar[1], Arunesh Sinha[1]

[1] Carnegie Mellon University
[2] Max Planck Institute for Software Systems

Abstract. Privacy has become a significant concern in modern society as personal information about individuals is increasingly collected, used, and shared, often using digital technologies, by a wide range of organizations. Certain information handling practices of organizations that monitor individuals' activities on the Web, data aggregation companies that compile massive databases of personal information, cell phone companies that collect and use location data about individuals, online social networks and search engines—while enabling useful services—have aroused much indignation and protest in the name of privacy. Similarly, as healthcare organizations are embracing electronic health record systems and patient portals to enable patients, employees, and business affiliates more efficient access to personal health information, there is trepidation that the privacy of patients may not be adequately protected if information handling practices are not carefully designed and enforced.

Given this state of affairs, it is very important to arrive at a general understanding of (a) why certain information handling practices arouse moral indignation, what practices or policies are appropriate in a given setting, and (b) how to represent and enforce such policies using information processing systems. This article summarizes progress on a research program driven by goal (b). We describe a *semantic model* and *logic of privacy* that formalizes privacy as a right to *appropriate flows of personal information*—a position taken by *contextual integrity*, a philosophical theory of privacy for answering questions of the form identified in (a). The logic is designed with the goal of enabling specification and enforcement of practical privacy policies. It has been used to develop the first complete formalization of two US privacy laws—the HIPAA Privacy Rule that prescribes and proscribes flows of personal health information, and the Gramm-Leach-Bliley Act that similarly governs flows of personal financial information. Observing that preventive access control mechanisms are not sufficient to enforce such privacy policies, we develop two complementary *audit mechanisms* for policy enforcement. These mechanisms enable auditing of practical privacy policies, including the entire HIPAA Privacy Rule. The article concludes with a vision for further research in this area.

1 Introduction

Privacy has become a significant concern in modern society as personal information about individuals is increasingly collected, used, and shared, often using digital technologies, by a wide range of organizations. Certain information handling practices of organizations that monitor individuals' activities on the Web, data aggregation companies that compile massive databases

* This work was partially supported by the U.S. Army Research Office contract "Perpetually Available and Secure Information Systems" (DAAD19-02-1-0389) to Carnegie Mellon CyLab, the NSF Science and Technology Center TRUST, the NSF CyberTrust grant "Privacy, Compliance and Information Risk in Complex Organizational Processes," the AFOSR MURI "Collaborative Policies and Assured Information Sharing," and HHS Grant no. HHS 90TR0003/01. Jeremiah Blocki and Henry DeYoung were also partially supported by NSF Graduate Fellowships. This work was mainly performed when Deepak Garg was at Carnegie Mellon University. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

of personal information, cell phone companies that collect and use location data about individuals, online social networks and search engines—while enabling useful services—have aroused much indignation and protest in the name of privacy (see, for example, a series of articles in the Wall Street Journal [51]). Similarly, as healthcare organizations are embracing electronic health record systems and patient portals to enable patients, employees, and business affiliates more efficient access to personal health information, there is trepidation that the privacy of patients may not be adequately protected if information handling practices are not carefully designed and enforced [29, 43, 49].

Given this state of affairs, it is very important to arrive at a general understanding of (a) why certain information handling practices arouse moral indignation, what practices or policies are appropriate in a given setting, and (b) how to represent and enforce such policies using information processing systems. This article summarizes progress on a research program driven by goal (b) [8, 9, 14, 22, 25]. The semantic model in this work is informed by *contextual integrity*—a philosphical theory of privacy for answering questions of the form identified in (a) [40]. Healthcare privacy has been a focus area of application for much of this work and consequently the examples in the paper are drawn from that domain. The article concludes with a vision for further research in this area.

Contextual Integrity The central thesis of contextual integrity is that *privacy is a right to appropriate flow of personal information*. The building blocks of this theory are *social contexts* and *context-relative informational norms*. A context captures the idea that people act and transact in society not simply as individuals in an undifferentiated social world, but as individuals in certain capacities (roles) in distinctive social contexts, such as healthcare, education, friendship and employment. Norms prescribe the flow of personal information in a given context, e.g., in a healthcare context a norm might prescribe flow of personal health information from a patient to a doctor and proscribe flows from the doctor to other parties who are not involved in providing treatment. Norms are a function of the following parameters: the respective roles of the sender, the subject, and the recipient of the information, the type of information, and the principle under which the information is sent to the recipient. Examples of transmission principles include confidentiality (prohibiting agents receiving the information from sharing it with others), reciprocity (requiring bi-directional information flow, e.g., in a friendship context), consent (requiring permission from the information subject before transmission), and notice (informing the information subject that a transmission has occured). When norms are contravened, people experience a violation of privacy. This theory has been used to explain why a number of technology-based systems and practices threaten privacy by violating entrenched informational norms. In addition, it provides a prescriptive method for determining appropriate norms for a context (see [40]).

Semantic Model and Logic of Privacy The idea that privacy expectations can be stated using context-relative informational norms is formalized in a *semantic model* and *logic of privacy* proposed by the first author and colleagues [8] and developed further in our follow-up work [22]. At a high-level, the model consists of a set of interacting agents in roles who perform actions involving personal information in a given context. For example, Alice (a patient) may send her personal health information to Bob (her doctor). Following the structure of context-relative informational norms, each transmission action is characterized by the roles of the sender, subject, receipient and the type of the information sent. Interactions among agents give rise to *traces* where each trace is an alternating sequence of states (capturing roles and knowledge of agents) and actions performed by agents that update state (e.g., an agent's knowledge may increase upon receiving a message or his role might change).

Transmission principles prescribe which traces respect privacy and which traces don't. While contextual integrity talks about transmission principles in the abstract, we require a precise logic for expressing them since our goal is to use information processing systems to check for violation of such principles. We were guided by two considerations in designing the logic: (a) *expressivity*—the logic should be able to represent practical privacy policies; and (b) *enforceability*—it should

be possible to provide automated support for checking whether traces satisfy policies expressed in the logic.

A logic of privacy that meets these goals is presented in our recent work [25]. We arrive at this enforceable logic by restricting the syntax of the expressive first-order logic we used in our earlier work to develop the first complete formalization of two US privacy laws—the HIPAA Privacy Rule for healthcare organizations and the Gramm-Leach-Bliley Act for financial institutions [22][1]. These comprehensive case studies shed light on common concepts that arise in transmission principles in practice—data attributes, dynamic roles, notice and consent (formalized as temporal properties), purposes of uses and disclosures, and principals' beliefs—as well as how individual transmission principles are composed in privacy policies[2]. We discuss these insights in Section 2 and their formalization in the semantic model and logic in Section 3.

Audit Mechanisms for Enforcing Privacy Policies We observe that access control mechanisms are not sufficient for enforcing all privacy policies because at run-time there may not be sufficient information to decide whether certain policy concepts (e.g., future obligations, purposes of uses and disclosures, and principals' beliefs) are satisfied or not. We therefore take the position that *audit mechanisms are essential for privacy policy enforcement.* The importance of audits has been recognized in the computer security literature. For example, Lampson [34] takes the position that audit logs that record relevant evidence during system execution can be used to detect violations of policy, establish accountability and punish the violators. More recently, Weitzner et al. [52] also recognize the importance of audit and accountability, and the inadequacy of preventive access control mechanisms as the sole basis for privacy protection in today's open information environment. However, while the principles of access control have been extensively studied, there is comparatively little work on the principles of audit. Our work is aimed at filling this gap. Specifically, we develop two complementary audit mechanisms for policy enforcement.

Our first insight is that *incomplete audit logs* provide a suitable abstraction to model situations (commonly encountered in practice) in which the log does not contain sufficient information to determine whether a policy is satisfied or violated, e.g., because of the policy concepts alluded to earlier—future obligations, purposes of uses and disclosures, and principals' beliefs. We formalize incomplete logs as partial structures that map each atomic formula to true, false or unknown. We design an algorithm, which we name reduce, to operate iteratively over such incomplete logs that evolve over time. In each iteration, reduce provably checks as much of the policy as possible over the current log and outputs a residual policy that can only be checked when the log is extended with additional information. We implement reduce and use it to check simulated audit logs for compliance with the entire HIPAA Privacy Rule. Our experimental results demonstrate that the algorithm scales to realistic audit logs. These results are summarized in Section 4 (see [25] for details).

Since privacy policies constrain flows of personal information based on subjective conditions (such as beliefs) that may not be mechanically checkable, reduce will output such conditions in the final residual policy leaving them to be checked by other means (e.g., by human auditors). The second audit algorithm, which we name Regret Minimizing Audits (RMA), learns from experience to provide operational guidance to human auditors about the coverage and frequency of auditing such subjective conditions. At a technical level, we formalize periodic audits in adversarial environments as an online learning problem over repeated games of imperfect information. The model takes pragmatic considerations into account, such as the periodic nature of audits, the audit budget and the loss that an organization incurs from privacy violations. RMA is a new regret minimization algorithm for this game model. These results are summarized in Section 5 (see [14] for details).

We conclude in Section 6 with a discussion of research directions in this area, including support for policy composition and evolution, formalizing seemingly subjective conditions (such

[1] This logic, in turn, generalizes the enforceable propositional temporal logic in [8].

[2] The model and logic supports information use actions in addition to transmission actions, so, strictly speaking, it can express policies that are more general than transmission principles.

as purposes and beliefs), and remaining challenges in the design of audit mechanisms for detecting policy violations, accountability mechanisms for appropriately assigning blame when violations are detected, and incentive mechanisms to deter adversaries from committing violations.

2 Concepts in Privacy Policies

Before discussing the formal details of our semantic model, logic of privacy and enforcement mechanisms, we provide an informal overview of the basic concepts in, and the overall structure of, practical privacy policies. Both the concepts and the overall structure are derived from a thorough analysis of all privacy requirements in the U.S. laws HIPAA and GLBA, which was started in [8] and completed in [22]. These concepts are the structure of the privacy laws; abstract data attributes of a transmission; the ability of agents, which we call principals, to dynamically alter the role in which they are active and roles to which they belong; the purpose of a transmission; agents' beliefs about their environment; and temporal conditions for both past provisions and future obligations. This overview simultaneously serves to justify the features of our logic of privacy, PrivacyLFP, which we formally describe in Section 3.

2.1 Structure of Privacy Policies

Positive and Negative Norms of Transmission In prior work, the first author and several colleagues applied the framework of contextual integrity to observe that privacy expectations inherent in laws like HIPAA and GLBA can, in general, be stated using context-relative informational norms of two kinds: *positive norms* ("may" conditions) and *negative norms* ("must" conditions) [8]. A transmission satisfies privacy expectations if any one positive norm and all negative norms applicable to the context of the transmission are satisfied. In subsequent work, several of the present authors demonstrated the entire privacy laws in HIPAA and GLBA can be formalized using this classification [22].

Practically, positive norms represent clauses of a law or policy which state that a transmission may occur *if* a condition is satisfied. For example, §164.506(c)(2) of HIPAA is a positive norm since it allows protected health information to be disclosed *if* the disclosure's purpose is treatment:

> "A covered entity may disclose protected health information for treatment activities of a health care provider."

In this way, positive norms capture the permitting clauses of a regulation. In general, out of all positive norms of a policy that apply to a disclosure, only one needs to be satisfied to deem the disclosure non-contradictory with the policy.

Negative norms represent policy clauses which state that a transmission may occur *only if* a condition is satisfied. For example, the core of HIPAA §164.508(a)(2) is a negative norm since it allows disclosure of psychotherapy notes *only if* it is authorized by the patient (modulo a few exceptions):

> "A covered entity must obtain an authorization for any use or disclosure of psychotherapy notes, except [. . .]."

Negative norms capture the denying clauses of a regulation because transmissions that do not satisfy the negative norms' conditions are disallowed. All negative norms of a policy that apply to a disclosure must be satisfied to prevent a violation of the policy when the disclosure occurs.

Exceptions to Norms of Transmission Both positive and negative norms may contain *exceptions*. For instance, HIPAA §164.508(a)(2) presented above as a negative norm has an exception elided by [. . .] earlier: "use [of the notes] by the originator of the psychotherapy notes for treatment." Taking this clause as a canonical example of a negative norm with an exception, we see that

exceptions provide a choice: a disclosure satisfies the policy if either there is evidence of the patient's authorization (thereby satisfying the norm's core), or there is evidence that the disclosure is a use by the notes' originator for treatment (thereby satisfying the norm's exception).

Similarly, positive norms can also have exceptions, though they have a different flavor. For example, §164.512(c)(1) of HIPAA allows a covered entity to disclose protected health information in reporting cases of abuse or domestic violence, but §164.512(c)(2) makes the exception that such disclosures are allowed only if the covered entity informs the victim of the report. These kind of exceptions simply refine the positive norm to a more specific set of conditions.

2.2 Common Concepts in Privacy Policies

Data Attributes Practical privacy policies define disclosure norms over abstract attributes of data such as "protected health information" or "psychotherapy notes". These abstract attributes of data often possess a hierarchical structure which the norms must respect. For example, psychotherapy notes are a particular type of protected health information, so every norm that prohibits flows of protected health information should also deny the same flows of psychotherapy notes (unless stated otherwise).

Dynamic Roles Just as policy norms refer to data attributes, but not to raw data, so also they usually refer to agents by their *roles*, not by their names. Thus, the legality of a particular disclosure usually depends on the role of the sender and recipient of the disclosure (e.g., a *psychiatrist* may legally disclose information to a *law enforcement official*, etc) and not their identities.

The roles held by an agent are not static; instead, they evolve over time. For example, §6803(a) of GLBA suggests that principals may become and cease to be customers of a financial institution, i.e., roles are *dynamic*:

> "At the time of establishing a customer relationship with a consumer and not less than annually during the continuation of such relationship, a financial institution shall provide a clear and conspicuous disclosure to such consumer [...], of such financial institution's policies and practices with respect to [disclosing nonpublic personal information]."

Moreover, this norm suggests that roles can be long-standing. The customer relationship is one such typically long-standing role since provisions for annual notices are required. But an agent is not active in the customer role at each moment of the several years during which he is in a customer relationship. Instead, he is variously active in the roles of parent, professor, patient, and, occasionally, customer during those years.

Past Provisions and Future Obligations Policy norms often refer to events at different points of time; allowing an agent to opt-in or opt-out of disclosures is a common example. For example, GLBA §6802(b)(1) requires a financial institution to allow opt-out:

> "A financial institution may not disclose nonpublic personal information to a nonaffiliated third party unless— [...] the consumer is given the opportunity, before the time that such information is initially disclosed, to direct that such information not be disclosed to such third party."

In other words, this norm makes the temporal requirement that, at some past time, the consumer was given the opportunity to opt-out of the disclosure and has not since exercised that opportunity. We use the term *provision* to refer to such requirements about past events.

Temporal (time-based) relations in privacy policies are not limited to provisions about the past. For example, HIPAA §164.510(a)(2) requires that covered entities provide an opportunity to opt-out of disclosures of directory information, with an exception for cases in which it is not practicable to provide that opportunity (e.g., when a patient is in a coma). However, if this exception is used, then §164.510(a)(3)(ii) demands that:

> "The covered health care provider must inform the individual and provide an opportunity to [opt-out of] uses or disclosures for directory purposes as required by paragraph (a)(2) of this section when it becomes practicable to do so."

This imposes a requirement for an event to occur in the future, though there is no concrete time limit since one cannot predict the time at which it will become practicable to provide an opportunity to opt-out. We use the term *obligation* to refer to such requirements for future events.

2.3 Subjective Concepts

The three concepts presented in Sections 2.2 are all *objective*, i.e, information required to evaluate the concepts in the context of a disclosure may possibly be available in mechanical form. For example, the attributes of data can be inferred by analyzing the data, the role of an agent at any given time will usually be available in a roles' database and the relative precedence of two events can be determined from the time stamps of their entries in their respective event logs. However, privacy policies also often depend on concepts that are *subjective*, i.e, have no representation in mechanical form. It is due to dependence on such concepts that enforcement of practical privacy policies cannot be completely automated and requires human intervention. In the following we discuss two such concepts, *viz.*, purposes of use and disclosure and individual beliefs.

Purposes of Uses and Disclosures Norms for use and disclosure of individual information often mention the purpose of the use or disclosure, as in §164.506(c)(2) of HIPAA:

> "A covered entity may disclose protected health information for treatment activities of a health care provider."

In general, determing whether such purpose requirements are respected may require human input[3] Like data attributes, purposes also obey a hierarchical structure, which must be reflected in PrivacyLFP. For example, the purpose of administering a blood test should be a refinement, or subpurpose, of the treatment purpose.

Agents' Beliefs Just as a transmission's intended purpose introduces an element of subjectivity, so do agents' beliefs and professional judgment. For example, HIPAA §164.512(f)(4) states:

> "A covered entity may disclose protected health information about an individual who has died to a law enforcement official for the purpose of alerting law enforcement of the death of the individual if the covered entity has a suspicion that such death may have resulted from criminal conduct."

The covered entity's belief that the death may have resulted from criminal conduct is absolutely crucial to the norm's meaning. Without this constraint, §164.512(f)(4) would permit a covered entity to disclose the protected health information of *any* deceased person to law enforcement officials.

3 Logic of Privacy and Its Semantic Model

Having informally described the structure of and common concepts in privacy policies, we present in this section a logic, PrivacyLFP, for representing privacy policies. We also present the logic's semantic model. Whereas the syntax of the logic is used to represent norms of privacy policies and their relation to each other, the semantic model formalizes relevant contextual information against which the truth or falsity of such norms is checked during enforcement. Such contextual information includes, but is not limited to, use and disclosure event logs, roles' databases and data attribute information.

[3] See Section 6 for a pointer to ongoing work on providing semantics to purpose requirements in privacy policies.

3.1 Overview

Technically, PrivacyLFP is first-order logic (predicate logic) with a slightly reduced syntax. The main syntactic categories in the logic are: 1) Terms, denoted t, which are symbolic representations of agents, data, attributes, roles, purposes, etc and over which variables $\boldsymbol{x}$ may range, 2) Predicates, denoted p, that represent relations between terms (e.g., Alice is a physician on 09/15/2011), and 3) Formulas, denoted φ, that are combinations of predicates using the usual connectives of logic — $\wedge$ (conjunction), $\vee$ (disjunction), $\supset$ (implication), $\forall \boldsymbol{x}.\varphi$ (for all instances of variables $\boldsymbol{x}$, φ), $\exists \boldsymbol{x}.\varphi$ (there is some instance of variables $\boldsymbol{x}$ such that φ), $\top$ (truth) and $\bot$ (falsity).

We represent both positive and negative norms (Section 2.1) as formulas, denoted φ^+ and φ^-, respectively. The exceptions of a norm are represented as subformulas of the formula representing the norm and are, therefore, part of the representation of the norm itself. If the positive norms applicable to a given disclosure are $\varphi_1^+, \ldots, \varphi_n^+$ whereas the negative norms applicable to the disclosure are $\varphi_1^-, \ldots, \varphi_m^-$, then the disclosure satisfies the policy if and only if the following formula is true: $(\varphi_1^+ \vee \ldots \vee \varphi_n^+) \wedge (\varphi_1^- \wedge \ldots \wedge \varphi_m^-)$. Following standard mathematics conventions, this formula is often abbreviated to $(\bigvee_i \varphi_i^+) \wedge (\bigwedge_j \varphi_j^-)$. In related work [22], several authors of this paper have formalized all norms from the HIPAA and GLBA Privacy Rules in this form. Although we do not discuss this formalization in detail here, an example representative of the formalization is shown later.

To represent the privacy policy concepts described in Sections 2.2 and 2.3, we stipulate a specific signature within PrivacyLFP that is inspired by prior work on the Logic of Privacy and Utility (LPU) [8]. Attributes of data (Section 2.2) are represented using symbolic terms (e.g., phi for protected health information). The hierarchy between data attributes is represented by a predicate $\texttt{attr\_in}(t_1, t_2)$, meaning that attribute t_1 is a subset of attribute t_2, e.g., $\texttt{attr\_in}(\text{medications}, \text{medical-history})$. We assume that each disclosed message is tagged (by its sender) with the attributes of information it carries and the predicate $\texttt{tagged}(m, q, t)$ means that the disclosed message m is tagged as carrying attribute t about agent q (e.g., a message may carry Alice's medical-history).

Similar to data attributes, role names (Section 2.2) are represented as symbolic terms, e.g, physician, covered-entity, etc. The relation between an agent p and its role r at time τ is represented by the formula $\texttt{inrole}(p, r, \tau)$. Including time in the relation allows accurate representation of dynamism in roles. For example, by including the time we allow for the possibility that (in our semantic model) $\texttt{inrole}(\text{Alice}, \text{physician}, 09/15/2011)$ is true but $\texttt{inrole}(\text{Alice}, \text{physician}, 09/16/2011)$ is not (Alice is a physician on 09/15/2011, but not on 09/16/2011).

Events such as use or disclosure of personal records are also represented by predicates called *event predicates*. For example, $\texttt{send}(p_1, p_2, m, \tau)$ means that agent p_1 sends message m to agent p_2 at time τ. A salient feature of PrivacyLFP is that it *requires* that every event predicate include a time argument like τ in $\texttt{send}(p_1, p_2, m, \tau)$. This time argument can be used to compare the order of occurence of two events using a relation $\tau \leq \tau'$ between time points. For example, the normative statement "if Alice sends a message to Bob, then Bob must later send a message to Alice" can be represented as: $\forall m, \tau.\ (\texttt{send}(\text{Alice}, \text{Bob}, m, \tau) \supset (\exists m', \tau'.\ ((\tau \leq \tau') \wedge \texttt{send}(\text{Bob}, \text{Alice}, m', \tau'))))$. We use this representation of time to encode both provisions and obligations in privacy policies (Section 2.2). Although PrivacyLFP does not include any explicit temporal operators (e.g., $\Diamond$ and $\Box$ [37]), it is more expressive than two common temporal logics — linear-time temporal logic (LTL) and timed propositional temporal logic (TPTL), as shown in related work [22].

Like data attributes, purposes of use and disclosure (Section 2.3) are represented by symbolic terms (e.g., "treatment", "healthcare", etc.). The predicate $\texttt{purp}(m, u)$ means that the purpose of disclosure of message m is u. Unlike all other predicates listed above, this predicate is *uninterpreted* — in any enforcement system only a human expert may decide whether or not the predicate holds for given m and u. (Later, we explain what uninterpreted means in our semantic model.)

Similar to disclosure purposes, agents' beliefs (Section 2.3) are also represented with uninterpreted predicates. To distinguish such predicates, their names begin with the prefix `believes-`, e.g., `believes-cause-of-death-is-crime`(p, q) may mean that agent p believes that agent q died due to a criminal act. Like the predicate `purp`(m, u), predicates beginning with the prefix `believes-` are also uninterpreted.

Example 1. We illustrate representation of privacy policies in PrivacyLFP with the following example that is motivated by similar requirements in HIPAA, but is simpler and serves as a good illustration.

> An entity (e.g., hospital or physician's office) may send an individual's protected health information (phi) to another entity only if the receiving entity is the individual's doctor and the purpose of the transmission is treatment, or the individual has previously consented to the transmission.

Observe that this policy contains two positive norms separated by the word "or" in the above quote. Using the descriptions of predicates presented earlier, this policy can be represented in PrivacyLFP as follows:

$$\begin{aligned}\varphi_{pol} = \forall p_1, p_2, m, q, t, \tau.\ & (\texttt{send}(p_1, p_2, m, \tau) \land \texttt{tagged}(m, q, t)) \\ & \supset \overline{\texttt{attr\_in}}(t, \text{phi}) \\ & \quad \lor (\texttt{inrole}(p_2, \texttt{doctorOf}(q), \tau) \land \texttt{purp}(m, \text{treatment})) \\ & \quad \lor \exists \tau'.\ (\tau' < \tau \land \texttt{consents}(q, \texttt{sendaction}(p_1, p_2, (q, t)), \tau'))\end{aligned}$$

The horizontal line over `attr_in` indicates negation: $\overline{\texttt{attr\_in}}(t, \text{phi})$ means that attribute t is *not* a subset of attribute phi. In words, the above formula φ_{pol} means that if entity p_1 sends to entity p_2 a message m at time τ and m is tagged as carrying attribute t of individual q, then either the attribute t is not a form of protected health information (so the policy does not apply) or the recipient p_2 is a doctor of q at time τ (atom `inrole`$(p_2, \texttt{doctorOf}(q), \tau)$) and the purpose of the disclosure is treatment, or q has consented to this transmission in the past (last line of φ_{pol}).

3.2 Syntax of the Logic of Privacy

Although we have discussed the syntax of PrivacyLFP and also illustrated it in the previous section, we summarize it below. The syntax deviates slightly from first-order logic because it includes a distinct category of formulas called restrictions (denoted c) and requires that all quantifiers contain these restrictions. The inclusion of restrictions is motivated by practical requirements: In the enforcement algorithm of Section 4, a quantifier's restriction allow us to finitely compute all *relevant* instances of the quantifier, which may otherwise be an infinite set. We also omit negation for technical convenience and assume that each predicate p has a dual $\overline{p}$ that behaves exactly like the negation of p. The negation $\overline{\varphi}$ of a formula φ can then be defined using De Morgan's laws, as usual.

$$\begin{array}{lll} \text{Terms} & t & ::= \ldots \\ \text{Atoms} & P & ::= p(t_1, \ldots, t_n) \\ \text{Formulas} & \varphi & ::= P \mid \top \mid \bot \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2 \mid \forall \boldsymbol{x}.(c \supset \varphi) \mid \exists \boldsymbol{x}.(c \land \varphi) \\ \text{Restrictions} & c & ::= P \mid \top \mid \bot \mid c_1 \land c_2 \mid c_1 \lor c_2 \mid \exists x.c \end{array}$$

3.3 Partial Structures and Semantics

Next, we describe the mathematical structures over which we interpret PrivacyLFP and its formal semantics. What mathematical structures are appropriate for interpreting PrivacyLFP? Like structures of first-order logic, we may expect the structures of PrivacyLFP to be abstractions of information about the truth and falsity of the logic's relations. Thus, a structure could be an

abstract data type encompassing all relevant event logs (to define predicates like `send`), roles' databases (to define predicates like `inrole`), and other auxiliary information about attribute hierarchies, etc.

However, unlike first-order logic where a structure maps each variable-free atom (relation) to either true or false, in PrivacyLFP we must also allow for the possibility that, in some cases, the information about the truth or falsity of a relation may be unavailable. This is primarily for three reasons. First, the logic includes subjective concepts like $\texttt{purp}(m, u)$ (message m is disclosed for purpose u), whose interpretation is unlikely to be available to any mechanized system of policy audit. We call such of lack of information *subjective incompleteness.* Second, the norms of a privacy policy may contain obligations that are to be satisfied in future; whether such obligations will hold or not cannot be determined during enforcement. We call this *future incompleteness.* Third, certain logs or databases may be unavailable at the time of policy enforcement, perhaps because they could not be integrated with the enforcement software. We call this *spatial incompleteness.*

To take into account all such potential incompleteness of information and to *force* ourselves to design enforcement mechanisms that take into account incompleteness, we interpret PrivacyLFP over *three-valued* structures that map each variable-free atom in the logic to one of three values: true (abbrev. `tt`), false (`ff`) or unknown (`uu`). Formally, a structure for PrivacyLFP (also called a *partial* structure) is a total map $\mathcal{L}$ from variable-free atoms of the logic to the set $\{\texttt{tt}, \texttt{ff}, \texttt{uu}\}$.

- Subjective incompleteness may be modeled in a partial structure $\mathcal{L}$ by mapping every predicate that describes a subjective relation to `uu`. For example, we may have $\mathcal{L}(\texttt{purp}(m, u)) = \texttt{uu}$ for every m and u. Predicates like `purp` may also be called uninterpreted.
- Future incompleteness may be modeled in a partial structure $\mathcal{L}$ using the time argument in every event predicate. For example, we may force $\mathcal{L}(\texttt{send}(p_1, p_2, m, \tau)) = \texttt{uu}$ whenever τ exceeds the time of audit.
- Spatial incompleteness may be modeled in a partial structure by mapping each predicate that is unavailable to `uu`. For instance, if the roles database is not available, then $\mathcal{L}(\texttt{inrole}(p, r, \tau)) = \texttt{uu}$ for every p, r, and τ.

In Section 4, we describe an audit-based method for enforcement of privacy policies using three-valued structures for interpreting policies. The method uniformly accounts for all these forms of incompleteness.

Semantics We formalize the semantics of logical formulas as the relation $\mathcal{L} \models \varphi$, read "$\varphi$ is true in the partial structure $\mathcal{L}$". Restrictions c are a subsyntax of formulas φ, so we do not define the relation separately for them. $\Xi[\boldsymbol{t}/\boldsymbol{x}]$ denotes substitution of terms $\boldsymbol{t}$ for variables $\boldsymbol{x}$ in the entity Ξ.

- $\mathcal{L} \models P$ iff $\mathcal{L}(P) = \texttt{tt}$
- $\mathcal{L} \models \top$
- $\mathcal{L} \models \varphi \wedge \psi$ iff $\mathcal{L} \models \varphi$ and $\mathcal{L} \models \psi$
- $\mathcal{L} \models \varphi \vee \psi$ iff $\mathcal{L} \models \varphi$ or $\mathcal{L} \models \psi$
- $\mathcal{L} \models \forall \boldsymbol{x}.(c \supset \varphi)$ iff for all $\boldsymbol{t}$ either $\mathcal{L} \models \overline{c}[\boldsymbol{t}/\boldsymbol{x}]$ or $\mathcal{L} \models \varphi[\boldsymbol{t}/\boldsymbol{x}]$
- $\mathcal{L} \models \exists \boldsymbol{x}.(c \wedge \varphi)$ iff there exists $\boldsymbol{t}$ such that $\mathcal{L} \models c[\boldsymbol{t}/\boldsymbol{x}]$ and $\mathcal{L} \models \varphi[\boldsymbol{t}/\boldsymbol{x}]$

For dual atoms, we define $\mathcal{L}(\overline{P}) = \overline{\mathcal{L}(P)}$, where $\overline{\texttt{tt}} = \texttt{ff}$, $\overline{\texttt{ff}} = \texttt{tt}$, and $\overline{\texttt{uu}} = \texttt{uu}$. We say that a formula φ is *false* on the structure $\mathcal{L}$ if $\mathcal{L} \models \overline{\varphi}$. The following two properties hold:

1. Consistency: A formula φ cannot be simultaneously true and false in the structure $\mathcal{L}$, i.e., either $\mathcal{L} \not\models \varphi$ or $\mathcal{L} \not\models \overline{\varphi}$
2. Incompleteness: A formula φ may be neither true nor false in a structure $\mathcal{L}$, i.e., $\mathcal{L} \not\models \varphi$ and $\mathcal{L} \not\models \overline{\varphi}$ may both hold.

Consistency means that a policy φ cannot be simultaneously violated and satisfied at the same time. Incompleteness means that there is a policy φ and a structure $\mathcal{L}$ such that it cannot be determined whether φ has been violated in $\mathcal{L}$ or not.

Structure Extension In practice, event logs and roles' databases evolve over time by gathering more information. This leads to a partial order, $\mathcal{L}_1 \leq \mathcal{L}_2$ on structures ($\mathcal{L}_2$ *extends* $\mathcal{L}_1$), meaning that $\mathcal{L}_2$ has more information than $\mathcal{L}_1$. Formally, $\mathcal{L}_1 \leq \mathcal{L}_2$ if for all variable-free atoms P, $\mathcal{L}_1(P) \in \{\mathtt{tt}, \mathtt{ff}\}$ implies $\mathcal{L}_2(P) = \mathcal{L}_1(P)$. Thus, as structures extend, the valuation of an atom may change from $\mathtt{uu}$ to either $\mathtt{tt}$ or $\mathtt{ff}$, but cannot change once it is either $\mathtt{tt}$ or $\mathtt{ff}$. The following property holds:

- Monotonicity: $\mathcal{L}_1 \leq \mathcal{L}_2$ and $\mathcal{L}_1 \models \varphi$ imply $\mathcal{L}_2 \models \varphi$.

Replacing φ with $\overline{\varphi}$, we also obtain that $\mathcal{L}_1 \leq \mathcal{L}_2$ and $\mathcal{L}_1 \models \overline{\varphi}$ imply $\mathcal{L}_2 \models \overline{\varphi}$. Hence, if $\mathcal{L}_1 \leq \mathcal{L}_2$ then $\mathcal{L}_2$ preserves both the $\mathcal{L}_1$-truth and $\mathcal{L}_1$-falsity of every formula φ.

4 Policy Audits over Incomplete Logs

In this section, we summarize an interactive algorithm for auditing system logs for privacy policy violation. To keep the presentation accessible, we present only the key ideas of our algorithm here and refer the reader to a technical paper [25] for details.

Ideally, we want our algorithm to answer the following question: Has a policy formula φ been violated in a (partial) structure $\mathcal{L}$? However, because we allow the structure to not have conclusive information about every atom, it is, in general, impossible to answer this question. Consequently, we take an *reduction-based approach*: Our algorithm implements a computable function reduce that takes as input a policy φ and a partial structure $\mathcal{L}$, and outputs a residual policy ψ that contains exactly the parts of φ that could not be verified due to lack of information in $\mathcal{L}$. Such an iteration is written $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$. If and when more information becomes available, extending $\mathcal{L}$ to $\mathcal{L}'$ ($\mathcal{L} \leq \mathcal{L}'$), another iteration of the algorithm can be used with inputs ψ and $\mathcal{L}'$ to obtain a new formula ψ'. This process can be continued until the output is either $\top$ (no violation) or $\bot$ (violation). A human auditor may augment the iterations by providing input about the truth or falsity of relevant uninterpreted atoms. By design, our algorithm satisfies three important properties:

- Termination: Each iteration terminates.
- Correctness: If $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$, then for all extensions $\mathcal{L}'$ of $\mathcal{L}$, $\mathcal{L}' \models \varphi$ iff $\mathcal{L}' \models \psi$.
- Minimality: If $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$, then an atom occurs in ψ only if it occurs in φ and its valuation on $\mathcal{L}$ is $\mathtt{uu}$.

Technical details The technically difficult part of reduce is its treatment of quantifiers over infinite domains. Consider, for instance, the behavior of an algorithm satisfying the above three properties on input $\forall x.\varphi$. Because the output must be minimal, in order to reduce $\forall x.\varphi$, a naive algorithm will instantiate x with each possible term and check the truth or falsity of φ for that instance on $\mathcal{L}$. This immediately leads to non-termination if the set of terms is infinite, which does happen for real policies (e.g., in Example 1, we quantify over messages m and time points τ, both of which are infinite sets).

Given the need for infinite domains, something intrinsic in quantification must limit the number of *relevant instances* of x that need to be checked to a finite number. To this end, we rely on the restrictions c in quantifiers, $\forall \boldsymbol{x}.(c \supset \varphi)$ and $\exists \boldsymbol{x}.(c \wedge \varphi)$, and use the technique of *mode analysis* from logic programming [2] to ensure that the restriction c has only a finite number of satisfying instances in any structure and that these instances are *computable*.

Briefly, mode analysis requires the policy designer to specify which argument positions of a predicate can be *computed finitely* from others. For instance, in Section 3.1 we assumed that the attributes of a message are written on it in machine-readable format and, hence, can be computed from the message. Denoting required inputs by $+$ and computable outputs by $-$, we may give the predicate $\mathtt{tagged}(m, q, t)$ the *mode* $\mathtt{purp}(+, -, -)$, meaning that from the input m, the outputs q, t can be computed. The mode $\mathtt{purp}(-, +, +)$ is incorrect because given a fixed second and third arguments (attribute), there may be an infinite number of first arguments (messages)

annotated with that attribute, so the latter set cannot be finitely computed. Similarly, if the predicate $\texttt{mult}(x, y, z)$ means that $x = yz$, where x, y, z are integers, then any of the modes $\texttt{mult}(+, +, -)$, $\texttt{mult}(-, +, +)$, and $\texttt{mult}(+, -, +)$ are okay, but $\texttt{mult}(-, -, +)$ is not. Given the mode information of all predicates in a policy, a static, linear-time check of the policy, called a *mode check*, ensures that there are only a finite number of instances of free variables that can satisfy a restriction c in the policy.

To actually *compute* the satisfying instances of a restriction, we define a function $\widehat{\texttt{sat}}(\mathcal{L}, c)$ that returns all substitutions σ for free variables of c such that $\mathcal{L} \models c\sigma$. This definition assumes a function $\texttt{sat}(\mathcal{L}, P)$ that returns all substitutions σ for free variables of P such that $\mathcal{L} \models P\sigma$ if all input positions in P are ground, which itself is implemented by looking up event logs or other databases, depending on the predicate in P.

Finally, the main audit function $\mathsf{reduce}(\mathcal{L}, \varphi)$ is defined by induction on φ, using $\widehat{\texttt{sat}}(\mathcal{L}, c)$ as a helper function when φ contains a top-level quantifier. The problematic case of the quantifiers is now easily dealt with: To reduce $\forall \boldsymbol{x}.(c \supset \varphi)$, we first invoke $\widehat{\texttt{sat}}(\mathcal{L}, c)$ to find all substitutions for $\boldsymbol{x}$ that satisfy c. Then, we recursively reduce φ after applying each such substitution and the output of reduce is the conjunction of these reducts. The reduction of $\exists \boldsymbol{x}.(c \wedge \varphi)$ is identical except that the output is the disjunction of the recursively obtained reducts.

Formal properties We formally prove that the $\mathsf{reduce}(\mathcal{L}, \varphi)$ is total for policies φ that pass our mode check, it is correct and minimal in the sense mentioned at the beginning of this section and that it uses space polynomial in the size of φ and runs in time polynomial in the size of $\mathcal{L}$.

Use for action-guidance Besides audit for policy violations, we expect that reduce can be used as an action-guidance tool, to inform an agent whether or not an action she is about to perform would violate the policy. To do this, reduce can be run on a *hypothetical* structure that includes all the audit information in the system and the action to be performed, and a formula that contains relevant norms from the privacy policy. The formula output by reduce would then be one of: (1) $\top$ (action will not violate the policy), (2) $\bot$ (action will violate the policy), or (3) another formula ψ which lists exactly those undischarged conditions that must hold for the policy to not be violated by the potential action; the agent may check those conditions manually before performing the action.

4.1 Related Work

Runtime Monitoring with Temporal Logic A lot of prior work addresses the problem of *runtime monitoring* of policies expressed in Linear Temporal Logic (LTL) [5, 7, 12, 44, 46, 47] and its extensions [7, 45, 46]. Although similar in the spirit of enforcing policies, the intended deployment of our work is different: We assume that system logs are accumulated independently and given to our algorithm, whereas an integral component of runtime monitoring is accumulation of system logs on the fly. Our assumption about the availability of system logs fits practical situations like health organizations, which collect transmission, disclosure and other logs to comply with regulations such as HIPAA even if no computerized policy enforcement mechanism is in place.

Comparing only the expressiveness of the logic, our work is more advanced than all existing work on policy enforcement. First, LTL can be encoded in our logic easily [22]. Second, we allow expressive quantification in our logic, whereas prior work is either limited to propositional logic [5, 44, 47], or, when quantifiers are considered, they are severely restricted [7, 45, 46]. A recent exception to such syntactic restrictions is the work of Basin et al. [12], to which we compare in detail below. Third, no prior work considers the possibility of incompleteness in structures, which our reduce algorithm takes into account.

Recent work by Basin et al. [12] considers runtime monitoring over an expressive fragment of Metric First-order Temporal Logic. Similar to our work, Basin et al. allow quantification over infinite domains, and use a form of mode analysis (called a safe-range analysis) to ensure finiteness during enforcement. However, Basin et al.'s mode analysis is weaker than ours; in

particular, it cannot relate the same variable in the input and output positions of two different conjuncts of a restriction and requires that each free variable appear in at least one predicate with a finite model. As a consequence, many practical policies (including examples from the HIPAA Privacy Rule) cannot be enforced in their framework, but can be enforced in ours (see [25] for additional details).

Formal Frameworks for Policy Audit Cederquist et al. [18] present a proof-based system for a-posteriori audit, where policy obligations are discharged by constructing formal proofs. The leaves of proofs are established from logs, but the audit process only checks that an obligation has been satisfied somewhere in the past. Further, there is no systematic mechanism to instantiate quantifiers in proofs. However, using connectives of linear logic, the mechanism admits policies that rely on use-once permissions.

Iterative Enforcement The idea of iteratively rewriting the policy over evolving logs has been considered previously [44, 47], but only for propositional logic where the absence of quantifiers simplifies the problem considerably. Bauer et al. [5] use a different approach for iterative enforcement: they convert an LTL formula with limited first-order quantification to a Büchi automaton and check whether the automaton accepts the input log. Further, they also use a three-valued semantic model similar to ours, but assume that logs record all information about past events (past-completeness). Three-valued structures have also been considered in work on generalized model checking [17, 27]. However, the problems addressed in that line of work are different; the objective there is to check whether there exist extensions of a given structure in which a formula is satisfied (or falsified).

Compliance Checking Barth et al. [8] present two formal definitions of compliance of an action with a policy, called strong and weak compliance. An action is *strongly compliant* with a policy given a trace if there exists an extension of the trace that contains the action and satisfies the policy. We do not consider strong compliance in this paper. An action is *weakly compliant* with a policy in Propositional LTL (PLTL) given a trace if the trace augmented with the action satisfies the present requirements of the policy. However, a weakly compliant action might incur unsatisfiable future requirements. The technical definition is stated in terms of a standard tableau construction for PLTL [37] that syntactically separates present and future requirements. Our correctness property for reduce generalizes weak compliance to a richer class of policies and structures: PLTL can be encoded in our policy logic, the residual formula generalizes future requirements, and past-complete traces are a special case of our partial structures.

In a related paper, Barth et al. [9] present an algorithm that examines audit logs to detect policy violations and identify agents to blame for policy violations. While our audit algorithm can be used to detect violations of a much richer class of policies than the propositional logics considered by Barth et al., it does not identify agents to be blamed for violations.

Lam et al. [33] represent policy requirements of a part of the HIPAA Privacy Rule in an extension of Prolog with stratified negation, called pLogic, and use it to implement a compliance checker for a medical messaging system. The compliance checker makes decisions about legitimacy of messages entering the system based on eight attributes attached to each message (such as its sender, intended recipient, subject, type of information and purpose). The prototype tool has a usable front-end and provides a useful interface for understanding what types of disclosures and uses of personal health information are permitted and forbidden by the HIPAA Privacy Rule. However, as recognized by the authors, the approach has certain limitations in demonstrating compliance with the HIPAA Privacy Rule. First, it does not support temporal conditions. While pLogic uses specialized predicates to capture that certain events happened in the past, it cannot represent future obligations needed to formalize many clauses in HIPAA. In contrast, our policy logic and the reduce algorithm handle temporal conditions, including real-time conditions. Second, reasoning in pLogic proceeds assuming that all asserted beliefs, purposes and types of information associated with messages are correct. In contrast, since reduce mines logs to determine truth values of atoms, it does not assume facts unless there is evidence in logs to

back them up. Typically, a purpose or belief will be taken as true only if a human auditor (or some other oracle) supplies evidence to that effect. Finally, our prototype implementation was evaluated with a formalization of the entire HIPAA Privacy Rule, whereas Lam et al. formalize only §§164.502, 164.506 and 164.510.

Policy Specification and Analysis Several variants of LTL have been used to *specify* the properties of programs, business processes and security and privacy policies [8, 11, 22, 26, 36]. The logic we use as well as the formalization of HIPAA used in our experiments are adapted from our prior work on the logic PrivacyLFP [22]. PrivacyLFP, in turn, draws inspiration from earlier work on the logic LPU [8]. However, PrivacyLFP is more expressive than LPU because it allows first-order quantification over infinite domains.

Further, several access-control models have extensions for specifying usage control and future obligations [13, 23, 28, 30, 39, 41, 42]. Some of these models assume a pre-defined notion of obligations [30, 39]. For instance, Irwin et al. [30] model obligations as tuples containing the subject of the obligation, the actions to be performed, the objects that are targets of the actions and the time frames of the obligations. Other models leave specifications for obligations abstract [13, 28, 42]. Such specific models and the ensuing policies can be encoded in our logic using quantifiers.

There also has been much work on analyzing the properties of policies represented in formal models. For instance, Ni et al. study the interaction between obligation and authorization [39], Irwin et al. have analyzed accountability problems with obligations [30], and Dougherty et al. have modeled the interaction between obligations and programs [23]. These methods are orthogonal to our objective of policy enforcement.

Finally, privacy languages such as EPAL [6] and privacyAPI [38] do not include obligations or temporal modalities as primitives, and are less expressive than our framework.

5 Periodic Audits with Imperfect Information

Since privacy policies constrain flows of personal information based on subjective conditions (such as purposes and beliefs) that may not be mechanically checkable, reduce will output such conditions in the final residual policy leaving them to be checked by other means (e.g., by human auditors). Recent studies have revealed that such subjective conditions are often violated in the real world in the healthcare domain; violations occur as employees access medical records of celebrities, family members, and neighbors motivated by general curiosity, financial gain, child custody lawsuits and other considerations that are not appropriate purposes for accessing patient records [29, 49]. In practice, organizations like hospitals conduct *ad hoc* audits in which the audit log, which records accesses and disclosures of personal information, is examined to determine whether personal information was appropriately handled.

In this section, we summarize an audit model and algorithm that can provide guidance to human auditors in this activity [14]. This work presents the first principled learning-theoretic foundation for audits of this form. Our first contribution is a *repeated game model* that captures the interaction between the defender (e.g., hospital auditors) and the adversary (e.g., hospital employees). The model includes a budget that constrains the number of actions that the defender can inspect thus reflecting the imperfect nature of audit-based enforcement, and a loss function that captures the economic impact of detected and missed violations on the organization. We assume that the adversary is worst-case as is standard in other areas of computer security. We also formulate a desirable property of the audit mechanism in this model based on the concept of *regret* in learning theory [16]. Our second contribution is a novel *audit mechanism* that provably minimizes regret for the defender. The mechanism learns from experience and provides operational guidance to the human auditor about which accesses to inspect and how many of the accesses to inspect. The regret bound is significantly better than prior results in the learning literature.

Mirroring the periodic nature of audits in practice, we use a repeated game model [24] that proceeds in rounds. A round represents an audit cycle and, depending on the application scenario, could be a day, a week or even a quarter.

Adversary Model In each round, the adversary performs a set of actions (e.g., accesses patient records) of which a subset violates policy. Actions are classified into types. For example, accessing celebrity records could be a different type of action from accessing non-celebrity records. The adversary capabilities are defined by parameters that impose upper bounds on the number of actions of each type that she can perform in any round. We place no additional restrictions on the adversary's behavior. In particular, we do not assume that the adversary violates policy following a fixed probability distribution; nor do we assume that she is rational. Furthermore, we assume that the adversary knows the defender's strategy (audit mechanism) and can adapt her strategy accordingly.

Defender Model In each round, the defender inspects a subset of actions of each type performed by the adversary. The defender has to take two competing factors into account. First, inspections incur cost. The defender has an audit budget that imposes upper bounds on how many actions of each type she can inspect. We assume that the cost of inspection increases linearly with the number of inspections. So, if the defender inspects fewer actions, she incurs lower cost. Note that, because the defender cannot know with certainty whether the actions not inspected were malicious or benign, this is a game of imperfect information [3]. Second, the defender suffers a loss in reputation for detected violations. The loss is higher for violations that are detected externally (e.g., by an Health and Human Services audit, or because information leaked as a result of the violation is publicized by the media) than those that are caught by the defender's audit mechanism, thus incentivizing the defender to inspect more actions.

In addition, the loss incurred from a detected violation depends on the type of violation. For example, inappropriate access of celebrities' patient records might cause higher loss to a hospital than inappropriate access of other patients' records. Also, to account for the evolution of public memory, we assume that violations detected in recent rounds cause greater loss than those detected in rounds farther in the past. The defender's audit mechanism has to take all these considerations into account in prescribing the number of actions of each type that should be inspected in a given round, keeping in mind that the defender is playing against the powerful strategic adversary described earlier.

Note that for adequate privacy protection, the economic and legal structure has to ensure that it is in the best interests of the organization to invest significant effort into auditing. Our abstraction of the reputation loss from policy violations that incentivizes organizations to audit can, in practice, be achieved through penalties imposed by government audits as well as through market forces, such as brand name erosion and lawsuits.

Regret Property We formulate a desirable property for the audit mechanism by adopting the concept of regret from online learning theory. The idea is to compare the loss incurred when the real defender plays according to the strategy prescribed by the audit mechanism to the loss incurred by a hypothetical defender with perfect knowledge of the number of violations of each type in each round. The hypothetical defender is allowed to pick a fixed strategy to play in each round that prescribes how many actions of each type to inspect. The *regret* of the real defender in hindsight is the difference between the loss of the hypothetical defender and the actual loss of the real defender averaged over all rounds of game play. We require that the regret of the audit mechanism quickly converge to a small value and, in particular, that it tends to zero as the number of rounds tends to infinity.

Intuitively, this definition captures the idea that although the defender does not know in advance how to allocate her audit budget to inspect different types of accesses (e.g., celebrity record accesses vs. non-celebrity record accesses), the recommendations from the audit mechanism should have the desirable property that over time the budget allocation comes close to the optimal fixed allocation. For example, if the best strategy is to allocate 40% of the budget to

inspect celebrity accesses and 60% to non-celebrity accesses, then the algorithm should quickly converge towards these values.

Audit Mechanism We develop a new audit mechanism that provably minimizes regret for the defender. The algorithm, which we name Regret Minimizing Audits (RMA), is efficient and can be used in practice. In each round of the game, the algorithm prescribes how many actions of each type the defender should inspect. It does so by maintaining weights for each possible defender action and picking an action with probability proportional to the weight of that action. The weights are updated based on a loss estimation function, which is computed from the observed loss in each round. Intuitively, the algorithm learns the optimal distribution over actions by increasing the weights of actions that yielded better payoff than the expected payoff of the current distribution and decreasing the weight of actions that yielded worse payoff.

Our main technical result is that the exact bound on regret for RMA is approximately $2\sqrt{2\frac{\ln N}{T}}$ where N is the number of possible defender actions and T is the number of rounds (audit cycles). This bound improves the best known bounds of $O\left(\frac{N^{1/3}\log N}{\sqrt[3]{T}}\right)$ for regret minimization over games of imperfect information. The main novelty is in the way we use a loss estimation function and characterize its properties to achieve the significantly better bounds. Specifically, RMA follows the structure of a regret minimization algorithm for perfect information games, but uses the estimated loss instead of the true loss to update the weights in each round. We define two properties of the loss estimation function—*accuracy* (capturing the idea that the expected error in loss estimation in each round is zero) and *independence* (capturing the idea that errors in loss estimation in each round are independent of the errors in other rounds)—and prove that any loss estimation function that satisfies these properties results in regret that is close to the regret from using an actual loss function. Thus, our bounds are of the same order as regret bounds for perfect information games. The better bounds are important from a practical standpoint because they imply that the algorithm converges to the optimal fixed strategy much faster.

5.1 Related Work

Zhao et al. [53] recognize that rigid access control can cause loss in productivity in certain types of organizations. They propose an access control regime that allows all access requests, but marks accesses not permitted by the policy for posthoc audit coupled with punishments for violating policy. They assume that the utility function for the organization and the employees are known and use a single shot game to analyze the optimal behavior of the players. Our approach of using a permissive access control policy coupled with audits is a similar idea. However, we consider a worst-case adversary (employee) because we believe that it is difficult to identify the exact incentives of the employee. We further recognize that the repeated nature of interaction in audits is naturally modeled as a repeated game rather than a one-shot game. Finally, we restrict the amount of audit inspections because of budgetary constraints. Thus, our game model is significantly more realistic than the model of Zhao et al. [53].

Cheng et al. [19, 20] also start from the observation that rigid access control is not desirable in many contexts. They propose a risk-based access control approach. Specifically, they allocate a risk budget to each agent, estimate the risk of allowing an access request, and permit an agent to access a resource if she can pay for the estimated risk of access from her budget. Further, they use metaheuristics such as genetic programming to dynamically change the security policy, i.e. change the risk associated with accesses dynamically. We believe that the above mechanism mitigates the problem of rigid access control in settings such as IT security risk management, but is not directly applicable for privacy protection in settings such as hospitals where denying access based on privacy risks could have negative consequences on the quality of care. Our approach to the problem is fundamentally different: we use a form of risk-based auditing instead of risk-based access control. Also, genetic programming is a metaheuristic, which is known to perform well empirically, but does not have theoretical guarantees [50]. In contrast, we provide mechanisms with provable guarantees. Indeed an interesting topic for future work is to investigate the use

of learning-theoretic techniques to dynamically adjust the risk associated with accesses in a principled manner.

Regret Minimization A regret minimization algorithm is a randomized algorithm for playing in a repeated game. Our algorithm RMA is based on the weighted majority algorithm [35] for regret minimization. The weighted majority maintains weights $w_{\boldsymbol{s}}$ for each of the N fixed actions of the defender. $w_{\boldsymbol{s}}^t$ is the weight of the expert before round t has been played. The weights determine a probability distribution over actions, $p_{\boldsymbol{s}}^t$ denotes the probability of playing $\boldsymbol{s}$ at time t. In any given round the algorithm attempts to learn the optimal distribution over actions by increasing the weights of experts that performed better than its current distribution and decreasing the weights of experts that performed worse.

Sleeping Experts In the setting of [35] all of the actions are available all of the time. However, we are working in the sleeping experts model where actions may not be available every round due to budget constraints. Informally, in the sleeping experts setting the regret of RMA with respect to a fixed action $\boldsymbol{s}$ in hindsight is the expected decrease in our total loss had we played $\boldsymbol{s}$ in each of the $T_{\boldsymbol{s}}$ rounds when $\boldsymbol{s}$ was available.

There are variations of the weighted majority algorithm that achieve low regret in the sleeping experts setting [15, 16]. These algorithms achieve average regret bounds:

$$\forall \boldsymbol{s}, \frac{\mathbf{Regret}\left(\mathsf{Alg}, \boldsymbol{s}\right)}{T_{\boldsymbol{s}}} = O\left(\frac{\sqrt{T \log N}}{T_{\boldsymbol{s}}}\right) .$$

In fact RMA is very similar to these algorithms. However, we are interested in finding exact (not asymptotic) bounds. We also have to deal with the imperfect information in our game.

Imperfect Information In order to update its weight after round t, the weighted majority algorithm needs to know the loss of every available defender action $\boldsymbol{s}$. Formally, the algorithm needs to know $\mathbf{L}^t(\boldsymbol{s})$ for each $\boldsymbol{s} \in \mathsf{AWAKE}^t$. However, we only observe an outcome $\mathbf{O}^t$, which allows us to compute

$$\mathbf{L}^t(\boldsymbol{s}^t) = \mathbf{R}(\mathbf{O}^t) - \boldsymbol{C} \cdot \boldsymbol{s}^t,$$

the loss for the particular action $\boldsymbol{s}^t$ played by the defender at time t. There are several existing algorithms for regret minimization in games with imperfect information [3, 4, 21, 54]. For example, [3] provides an average regret bound of

$$\forall \boldsymbol{s}, \frac{\mathbf{Regret}(\mathsf{Alg}, \boldsymbol{s})}{T} = O\left(\frac{N^{1/3} \log N}{\sqrt[3]{T}}\right) .$$

It is acceptable to have $\log N$ in the numerator, but the $N^{1/3}$ term will make the algorithm impractical in our setting. The average regret still does tend to 0 as $T \to \infty$, but the rate of convergence is much slower compared to the case when only $\log N$ is present in the numerator. Other algorithms [4, 21, 54] improve this bound slightly, but we still have the $N^{1/3}$ term in the numerator. Furthermore, [3] assumes that each action $\boldsymbol{s}$ is available in every round. There are algorithms that deal with sleeping experts in repeated games with imperfect information, but the convergence bounds get even worse.

Regret minimization techniques have previously been applied in computer security by Barth et al. [10]. However, that paper addresses a different problem. They show that reactive security is not worse than proactive security in the long run. They propose a regret minimizing algorithm (reactive security) for allocation of budget in each round so that the attacker's "return on attack" does not differ much from the case when a fixed allocation (proactive security) is chosen. Their algorithm is not suitable for our audit setting due to imperfect information and sleeping experts. In their work, the defender learns the attack path played by the adversary after each round, and by extension has perfect knowledge of the loss function for that round. By contrast, RMA must work in the imperfect information setting. Also, their model considers unknown attack paths that get discovered over time. This is a special subcase of the sleeping experts setting, where an expert is awake in every round after she wakes up. They extend the multiplicative weight update

algorithm [35] to handle the special case. In our setting experts may be available in one round and unavailable in next. RMA was designed to work in this more general setting.

6 Research Directions

We describe below directions for further research in this area, including support for policy composition and evolution, formalizing seemingly subjective conditions (such as purposes and beliefs), and remaining challenges in the design of audit mechanisms for detecting policy violations, accountability mechanisms for appropriately assigning blame when violations are detected, and incentive mechanisms to deter adversaries from committing violations.

While our work so far has focused on studying a single policy in one context (e.g., HIPAA for healthcare), it would be interesting to study situations where multiple policies from possibly different contexts may be relevant to the disclosure and use of personal information. For example, in the US, transmission of personal health information is governed not only by the HIPAA Privacy Rule, but also by state privacy laws. In Europe, in addition to the EU Directive, member states have their own privacy laws. A natural set of research questions arises in this setting: How should multiple policies from possibly different contexts be composed? How should conflicts be resolved? Is it possible to develop specification and enforcement techniques that are compositional? Is it possible to deal with policies (e.g., laws) that evolve over time in an incremental manner rather than requiring a significant rewrite?

As noted earlier, privacy policies often contain obligations based on beliefs or professional judgment of agents (Section 2.3). Such obligations are subjective and, in general, cannot be verified automatically by a computer system without human input. One natural question is how to provide human input about a relevant belief to a computerized audit mechanism. One possibility, already developed in our implementation of the reduce algorithm of Section 4, is to simply allow a user to mark a subjective concept as either true or false. Another possibility, which we plan to investigate in future, is to use logical rules to define that a belief is justified if certain principals support certain statements. We plan to use the connective A says φ (principal A supports statement φ) from authorization logics to represent statements made by principals and signed certificates to evidence such statements [1]. For example, the rule $(P$ says `injured-by-crime`$(P)) \supset$ `believes-injured-by-crime`(Q, P) may mean that if principal P says that she was injured by a crime then it is justified for another individual Q to believe that this is the case, and a certificate signed by Alice's private key and containing the statement `injured-by-crime`(Alice) could be evidence for the formula (Alice says `injured-by-crime`(Alice)). Certificates necessary to justify relevant beliefs may be collected by an audit system and used to discharge obligations about beliefs automatically.

An approach to semantics and enforcement of privacy policies that place requirements on the *purposes* for which a governed entity may use personal information is outlined in a recent article by the first author and colleagues [48]. The paper presents a semantics for purpose requirements using a formal model based on planning. Specifically, the model is used to formalize when a sequence of actions is *only for* or *not for* a purpose. This semantics serves as a basis for an algorithm for automated auditing of purpose requirements in privacy policies. The algorithm takes as input an audit log and a description of an environment model that the auditee uses in the planning process.

In addition to audit mechanisms that detect violations of policy, an important research direction is developing a rigorous foundation for *accountability* and associated mechanisms that correctly blame agents responsible for violations. While the importance of accountability has been recognized in the literature[34, 52], there has not been much technical work on accountability (see [9, 31, 32] for some exceptions).

Also, while our work on regret minimizing audits makes no assumptions about the *incentives* of adversaries, a worthwhile research direction is to design mechanisms that use partial knowledge of the incentives of adversaries to deter them from committing violations. We expect that a

combination of techniques from game theory and learning theory could be leveraged to make progress on this problem.

Finally, while our work has focused on the application domain of healthcare privacy, an exploration of other domains in which these enforcement mechanisms could be used would be interesting. Specifically, it would be worthwhile to investigate whether privacy protection policies adopted by financial institutions, web services providers (e.g., Google, Microsoft, Amazon) and online social networks (e.g., Facebook) can be enforced by using and adapting the kinds of mechanisms being developed in this work.

References

[1] Abadi, M., Burrows, M., Lampson, B. W., and Plotkin, G. D. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst. 15*, 4 (1993), 706–734.

[2] Apt, K. R., and Marchiori, E. Reasoning about Prolog programs: From modes through types to assertions. *Formal Aspects of Computing 6*, 6 (1994), 743–765.

[3] Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing 32*, 1 (2003), 48–77.

[4] Awerbuch, B., and Kleinberg, R. Online linear optimization and adaptive routing. *Journal of Computer and System Sciences 74*, 1 (2008), 97–114.

[5] Baader, F., Bauer, A., and Lippmann, M. Runtime verification using a temporal description logic. In *Proceedings of the 7th International Conference on Frontiers of Combining Systems (FroCos)* (2009), pp. 149–164.

[6] Backes, M., Pfitzmann, B., and Schunter, M. A toolkit for managing enterprise privacy policies. In *Proceedings of the 8th European Symposium on Research in Computer Security (ESORICS)* (2003), LNCS 2808, pp. 101–119.

[7] Barringer, H., Goldberg, A., Havelund, K., and Sen, K. Rule-based runtime verification. In *Proceedings of the 5th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)* (2004), pp. 44–57.

[8] Barth, A., Datta, A., Mitchell, J. C., and Nissenbaum, H. Privacy and contextual integrity: Framework and applications. In *Proceedings of the 27th IEEE Symposium on Security and Privacy (Oakland)* (2006), pp. 184–198.

[9] Barth, A., Datta, A., Mitchell, J. C., and Sundaram, S. Privacy and utility in business processes. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF)* (2007), pp. 279–294.

[10] Barth, A., Rubinstein, B., Sundararajan, M., Mitchell, J., Song, D., and Bartlett, P. A Learning-Based Approach to Reactive Security. *Financial Cryptography and Data Security* (2010), 192–206.

[11] Basin, D., Klaedtke, F., and Müller, S. Monitoring security policies with metric first-order temporal logic. In *Proceeding of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT)* (2010), pp. 23–34.

[12] Basin, D. A., Klaedtke, F., and Müller, S. Policy monitoring in first-order temporal logic. In *Proceedings of the 22nd International Conference on Computer Aided Verification (CAV)* (2010), pp. 1–18.

[13] Bettini, C., Jajodia, S., Wang, X. S., and Wijesekera, D. Provisions and obligations in policy rule management. *Journal of Network and Systems Management 11* (2003), 351–372.

[14] Blocki, J., Christin, N., Datta, A., and Sinha, A. Regret minimizing audits: A learning-theoretic basis for privacy protection. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium (CSF)* (2011), pp. 312–327.

[15] Blum, A., and Mansour, Y. From external to internal regret. In *COLT* (2005), pp. 621–636.

[16] Blum, A., and Mansour, Y. Learning, regret minimization, and equilibria. *Algorithmic Game Theory* (2007), 79–102.

[17] Bruns, G., and Godefroid, P. Generalized model checking: Reasoning about partial state spaces. In *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR)* (2000), pp. 168–182.

[18] Cederquist, J. G., Corin, R., Dekker, M. A. C., Etalle, S., den Hartog, J. I., and Lenzini, G. Audit-based compliance control. *International Journal of Information Security 6*, 2 (2007), 133–151.

[19] Cheng, P.-C., and Rohatgi, P. IT Security as Risk Management: A Research Perspective. *IBM Research Report RC24529* (April 2008).

[20] Cheng, P.-C., Rohatgi, P., Keser, C., Karger, P. A., Wagner, G. M., and Reninger, A. S. Fuzzy Multi-Level Security : An Experiment on Quantified Risk-Adaptive Access Control. In *Proceedings of the IEEE Symposium on Security and Privacy* (2007).

[21] Dani, V., and Hayes, T. Robbing the bandit: Less regret in online geometric optimization against an adaptive adversary. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm* (2006), ACM, p. 943.

[22] DeYoung, H., Garg, D., Jia, L., Kaynar, D., and Datta, A. Experiences in the logical specification of the HIPAA and GLBA privacy laws. In *Proceedings of the 9th Annual ACM Workshop on Privacy in the Electronic Society (WPES)* (2010). Full version: Carnegie Mellon University Technical Report CMU-CyLab-10-007.

[23] Dougherty, D. J., Fisler, K., and Krishnamurthi, S. Obligations and their interaction with programs. In *Proceedings of the 12th European Symposium on Research in Computer Security (ESORICS)* (2007), pp. 375–389.

[24] Fudenberg, D., and Tirole, J. *Game theory.* MIT Press, 1991.

[25] Garg, D., Jia, L., and Datta, A. Policy auditing over incomplete logs: Theory, implementation and applications. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)* (2011).

[26] Giblin, C., Liu, A. Y., Müller, S., Pfitzmann, B., and Zhou, X. Regulations expressed as logical models (REALM). In *Proceeding of the 18th Annual Conference on Legal Knowledge and Information Systems (JURIX)* (2005), pp. 37–48.

[27] Godefroid, P., and Huth, M. Model checking vs. generalized model checking: Semantic minimizations for temporal logics. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS)* (2005), pp. 158–167.

[28] Hilty, M., Basin, D. A., and Pretschner, A. On obligations. In *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS)* (2005), pp. 98–117.

[29] Hulme, G. Steady Bleed: State of HealthCare Data Breaches. InformationWeek, September 2010. Available at http://www.informationweek.com/blog/healthcare/229200720.

[30] Irwin, K., Yu, T., and Winsborough, W. H. On the modeling and analysis of obligations. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)* (2006), pp. 134–143.

[31] Jagadeesan, R., Jeffrey, A., Pitcher, C., and Riely, J. Towards a theory of accountability and audit. In *ESORICS* (2009), pp. 152–167.

[32] Küsters, R., Truderung, T., and Vogt, A. Accountability: definition and relationship to verifiability. In *ACM Conference on Computer and Communications Security* (2010), pp. 526–535.

[33] Lam, P. E., Mitchell, J. C., and Sundaram, S. A formalization of HIPAA for a medical messaging system. In *Proceedings of the 6th International Conference on Trust, Privacy and Security in Digital Business (TrustBus)* (2009), pp. 73–85.

[34] Lampson, B. W. Computer security in the real world. *IEEE Computer 37*, 6 (2004), 37–46.

[35] Littlestone, N., and Warmuth, M. K. The weighted majority algorithm. *Inf. Comput. 108*, 2 (1994), 212–261.

[36] Liu, Y., Müller, S., and Xu, K. A static compliance-checking framework for business process models. *IBM Systems Journal 46* (2007), 335–361.

[37] Manna, Z., and Pnueli, A. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, 1995.

[38] May, M. J., Gunter, C. A., and Lee, I. Privacy APIs: Access control techniques to analyze and verify legal privacy policies. In *Proceedings of the 19th IEEE Workshop on Computer Security Foundations (CSFW)* (2006), pp. 85–97.

[39] Ni, Q., Bertino, E., and Lobo, J. An obligation model bridging access control policies and privacy policies. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies (SACMAT)* (2008), pp. 133–142.

[40] Nissenbaum, H. *Privacy in Context: Technology, Policy, and the Integrity of Social Life*. Stanford University Press, 2010.

[41] OASIS XACML Committee. Extensible access control markup language (XACML) v2.0, 2004. Available at http://www.oasis-open.org/specs/#xacmlv2.0.

[42] Park, J., and Sandhu, R. Towards usage control models: beyond traditional access control. In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT)* (2002), pp. 57–64.

[43] Robertson, J. New data spill shows risk of online health records. Yahoo News, August 2011. Available at http://news.yahoo.com/data-spill-shows-risk-online-health-records-120743449.html.

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

[45] Roger, M., and Goubault-Larrecq, J. Log auditing through model-checking. In *Proceedings of the 14th IEEE Workshop on Computer Security Foundations (CSF)* (2001), pp. 220–236.

[46] Sokolsky, O., Sammapun, U., Lee, I., and Kim, J. Run-time checking of dynamic properties. *Electronic Notes in Theoretical Computer Science 144* (2006), 91–108.

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

[48] Tschantz, M. C., Datta, A., and Wing, J. On the semantics of purpose requirements in privacy policies. Tech. Rep. CMU-CS-11-102, Carnegie Mellon University, 2010.

[49] US Health and Human Services. HIPAA enforcement, Accessed November 19,2010. Available at http://www.hhs.gov/ocr/privacy/hipaa/enforcement/index.html.

[50] Vose, M. D., Wright, A. H., and Rowe, J. E. Implicit parallelism. In *IN GECCO (2003)* (2003), pp. 1505–1517.

[51] Wall Street Journal. What they know, Accessed on September 8, 2011. Available at http://online.wsj.com/public/page/what-they-know-digital-privacy.html.

[52] Weitzner, D. J., Abelson, H., Berners-Lee, T., Feigenbaum, J., Hendler, J. A., and Sussman, G. J. Information accountability. *Commun. ACM 51*, 6 (2008), 82–87.

[53] Zhao, X., and Johnson, M. E. Access governance: Flexibility with escalation and audit. In *HICSS* (2010), pp. 1–13.

[54] Zinkevich, M., Johanson, M., Bowling, M., and Piccione, C. Regret minimization in games with incomplete information. *Advances in Neural Information Processing Systems 20* (2008), 1729–1736.