DESIGN AND ANALYSIS OF PRIVACY POLICIES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Adam Barth
August 2008

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(John C. Mitchell) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Dan Boneh)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Rajeev Motwani)

Approved for the University Committee on Graduate Studies.

Acknowledgments

Many thanks are in order. Although some might consider it cliché, I must thank my parents, Jeffrey and Mary Barth, for motivating me to write this thesis and for their careful, thoughtful comments about its contents. My advisor, John C. Mitchell, of course, deserves a great deal of thanks for, without his tutelage, this thesis would not have come to be. I also want to thank Dan Boneh, who often served as my second, unofficial advisor (or wartime consul, so to speak), for offering his insight, good humor, and diplomatic skills. Further, I would be remiss if I did not thank Anupam Datta, John C. Mitchell, Helen Nissenbaum, Sharada Sundaram, and Justin Rosenstein, who were all instrumental in the creation of this thesis by co-authoring the papers in which many of the ideas in this thesis were first published. My reading committee, John C. Mitchell, Dan Boneh, and Rajeev Motwani, have earned my thanks for reading this thesis all the way to the end and staking their reputation on the adequacy of my work. I thank my orals committee, Ron Kasznik, John C. Mitchell, Dan Boneh, Rajeev Motwani, and Monica Lam, for publicly interrogating me about my thesis in front of my friends, family, and colleagues so that I might offer my defense. Additionally, I thank John C. Mitchell, Dan Boneh, and Tim Roughgarden for examining whether I was qualified to begin writing this thesis. Kathi DiTommaso deserves thanks for keeping "the checklist" and ensuring that a sufficient number of check marks appeared on the list every year. Finally, I wish to thank Aaron Bradley, César Sánchez, and Matteo Slanina for helpful discussions about temporal logic.

Abstract

Organizations, such as hospitals and financial institutions, that use privacy-sensitive information face the challenge of complying with privacy regulations and their own privacy policies. These regulations and policies are often written in natural language (or legalese), making it difficult for information systems to aid in assuring compliance. In this thesis, we propose a formal language for expressing and reasoning about privacy regulations and policies.

Other researchers have proposed other privacy languages, but these languages suffer semantic anomalies due to their handling of the "data hierarchy," the relation between different attributes about the same individual. We analyze a number of examples of such anomalies in the Platform for Privacy Preferences and in the Enterprise Privacy Authorization Language and lay out a set of criteria for evaluating privacy languages.

We present our language, the Logic of Privacy and Utility, which is based on Contextual Integrity, a theory of privacy expectations from the literatures on law and public policy. Our language formalizes a portion of Contextual Integrity as a concurrent game structure of communicating agents. We then use a fragment of the Alternating-time Temporal Logic of this model as our privacy language and identify specific syntactic forms for expressing the norms of Contextual Integrity.

We evaluate the privacy features of the language in three ways. First, we present theorems about the complexity of combination and compliance, distinguishing between weak compliance (which does not consider the feasibility of future obligations)

and strong compliance (which guarantees that agents can discharge their future obligations). Second, we compare the language with other approaches to codifying privacy policies, finding that our language generalizes a number of other approaches. Third, we show that the language is capable of expressing the privacy requirements from a number of privacy regulations, including the Health Insurance Portability and Accountability Act.

To evaluate the utility features, those features that aid in reasoning about the usefulness of various data practices, we offer a theory of organizational workflows, also known as business processes. In this setting, we examine the trade-offs between privacy and utility in workflow design (including notions of "minimum necessary" information disclosure) and offer practical algorithms for auditing workflow execution to discover agents who both violate their workflow responsibilities and cause the organization to violate its privacy policy.

Contents

Chapter 1

Introduction

In the past few decades, we have seen a radical intensification in the social practices of gathering, storing, manipulating, and sharing information about people (henceforth, "personal information"). In many instances, new practices have aroused suspicion, indignation, and protest not only among legal experts, social critics, and privacy advocates, but also in the popular media and among the general public. Recent controversies range from the introduction of Caller ID to Lotus Marketplace Households and EZ Pass, from Carnivore and "total information awareness" to Internet cookies and online profiling. This societal awareness [1] has lead to privacy becoming an important business concern in health care, financial services, and other organizations.

Hospitals, clinics, banks, credit card clearing houses, customer support centers, and academic institutions, among others, all maintain databases with sensitive information. These databases are used regularly by employees to carry out business-critical tasks. Organizations that collect and use personal information face the growing challenge of conducting their business effectively while managing privacy risks and compliance requirements. The risks are very real, with the theft of 26 million veteran records in May 2006 demonstrating how easily sensitive information can fall into unauthorized hands [24]. In the United States, privacy regulations, such as the Health Insurance Portability and Accountability Act (HIPAA) [54] for the health care sector, the Children's Online Privacy Protection Act (COPPA) [35] for e-business, and the Gramm–Leach–Bliley Act (GLBA) [36, 34] for financial institutions, have spurred

many businesses, including 68% of the Direct Marketing Association member companies as of 2001 [28], to appoint Chief Privacy Officers whose primary job is being responsible for privacy issues and policies.

One of the biggest problems that privacy-sensitive organizations face is designing their internal activities and information practices to simultaneously serve their customers effectively and manage risks from disclosure of sensitive information. This fundamental problem arises in hospitals and clinics, where personal health information must be used to provide effective health care, but must also be protected from indiscriminate sharing to respect the privacy of patients—a requirement made more precise by HIPAA. Financial institutions use sensitive financial information to decide whether to grant loans, for example, and suffer direct loss and brand erosion if sensitive information is lost. Retail enterprises use credit card details in resolving charge-back disputes (where the privacy concerns are exacerbated [9] by the common practice of outsourcing this task). College admissions officers review confidential letters of recommendation and transcripts.

To manage these risks and ameliorate consumer concerns, these organizations have developed privacy policies and business processes that govern their use of confidential information. However, these privacy policies are often represented in legalistic free text [40] and are difficult to integrate directly into the organization's business processes and information systems, leaving executive and consumers to wonder whether the organization actually complies with its own privacy policy. For these policies to be effective, the organization must carefully design the way it processes and uses information to balance the competing goals of privacy and the usefulness, or utility, of the business process.

Business process designs involve instructing individuals how and when to access and use information, coupled with access and use policies embedded in information processing systems. Because considering utility or privacy alone does not provide enough information to make meaningful management decisions, our goal is to develop a framework and model for designing, evaluating, and auditing business processes to achieve utility goals while minimizing privacy risks, with particular attention to the requirements of health care providers and financial institutions.

1.1 Privacy

Although there are philosophical theories of the nature and value of privacy, these tend to offer an account of what privacy is—say, control over information about oneself—and might explain why it ought to be valued and protected in liberal democracies. In contrast, the framework of *contextual integrity* has arisen in recent years to provide guidance on how to respond to conflicts between values and interests and to provide a systematic setting for understanding privacy expectations and the reasons that some events cause moral indignation [52, 56]. The central tenant of contextual integrity is that people interact in society not simply as individuals in an undifferentiated social world, but as individuals in particular capacities or roles, in distinctive social contexts (e.g., health care or banking). Our logic of privacy and utility is a formal framework for expressing privacy expectations and privacy practices inspired by contextual integrity.

We begin with a simple model of the transmission of personal information, containing communications such as "Alice sends Bob a particular type of information about Carol," and use first-order temporal logic for expressing and reasoning about which communications are and are not appropriate. The central concepts drawn from contextual integrity include contexts, roles, and a focus on the type of information transmitted (Carol's height) rather than specifics of the data (Carol is 5'10" tall). Roles within contexts are used to express that communication that is perfectly acceptable between a psychiatrist and patient is completely unacceptable between a human resource specialist and a job applicant. Temporal logic with *past* and *future* operators is used to say, for example, that particular information may be disclosed only if the subject mentioned has previously given permission or that if particular information is made public, notification must be sent to the concerned party. Although contextual integrity was developed to support specific, substantive philosophical and legal positions, our goal is to formalize concepts from contextual integrity so that privacy guidelines, policies, and expectations can be stated precisely, compared, and enforced by an information processing system.

We define two kinds of norms, which we call *positive* and *negative*, as temporal logic formulas of two particular forms. These two kinds of norms generalize "allow" and "deny" rules found in traditional access control languages to settings with temporal conditions. A positive norm permits some communication *if* its temporal condition is satisfied, whereas a negative norm forbids some communication *unless* its temporal condition is satisfied. We give these formulas semantics by interpreting them over a model of communicating agents. The privacy fragment of the logic is contained in Linear Temporal Logic (see, for example, [48]) and concerns trace histories of agent communications. One agent communicates with another agent by sending information about a subject. Our model of "information" includes a relation that lets agents combine messages to compute additional information about the subject (e.g., computing a postal code from a postal address), elucidating the notion of a "data hierarchy" found in the Platform for Privacy Preferences (P3P) [26] and the Enterprise Privacy Authorization Language (EPAL) [41]. To illustrate the expressiveness of this framework and explain its use, we show how to capture privacy provisions of HIPAA, COPPA, and GLBA as combinations of positive and negative norms with temporal conditions.

Our current framework makes two simplifying assumptions: norms are based only on the type of information communicated and information is assumed to describe an individual rather than a group of individuals. For example, we can easily express that it is acceptable for a physician to record particular types of information, but it is outside the scope of our current language to say that the average salary of bank managers can be released only if it does not identify a particular individual's salary. We believe it will be fruitful to develop precise connections with research on data privacy and aggregation in the future, but for simplicity we do not consider these extensions at present.

Like much of the work on access control and privacy languages in the computer security community, we express privacy policies in a formal logic and relate issues of compliance and refinement to the logical concepts of satisfiability and entailment. In addition to evaluating the expressive power of our language, we present specific technical results characterizing policy consistency, entailment, and compliance, leveraging

the existing literature on Linear Temporal Logic (LTL). Entailment is key to understanding how to combine policies and how to compare one policy, such as HIPAA, with another policy, such as the specific privacy practices of a clinic and hospital. Previous work on privacy languages used a complex lattice-based definition of entailment. In our model, entailment is captured as standard logical implication, letting us define and compute policy combination through the usual logical operations of conjunction and disjunction and without reference to complex objects, such as data hierarchies, that have been responsible for the anomalous semantics of combination in earlier languages.

1.2 Utility

Analyzing an organization's data practices in a privacy-only framework often leads to advocating for stricter control of information. A privacy-only analysis fails to consider the useful purpose the data practices serve for the organization. For example, in a hospital, nurses and doctors need to share sensitive information about patients in order to provide health care services. This information sharing is useful, has *utility*, because it furthers the purpose of the health care context, i.e., promoting health.

By formulating an organization's data practices as a business process, or *workflow*, we are able to formalize the notion of utility in a unified framework with privacy, letting us examine the trade-offs between privacy and utility. Our approach expands the LTL fragment of the logic of privacy and utility to Alternating-time Temporal Logic (ATL*) by including a strategy quantifier. The strategy quantifier lets us reason about what useful outcomes individuals interacting via the workflow can achieve. We illustrate these concepts using MyHealth@Vanderbilt [69], a web-based patient portal built and used at the Vanderbilt Medical Center. Examining the MyHealth portal led to many insights captured in our general theory.

The individuals in MyHealth act as patients, doctors, nurses, or secretaries, according to a specific workflow for scheduling appointments, viewing lab results, and asking and answering heath questions. One utility goal for MyHealth is to respond to health questions from patients. The MyHealth workflow also has privacy goals,

which we express using positive and negative norms that specify the conditions under which personal information can be communicated from one party to another. For example, one of MyHealth's privacy goals is to restrict health information to doctors and nurses, the health care providers.

Using a model of *actions* that transmit personal information from a sender in one role to a receiver in a possibly different role, agents may accumulate and send different types of personal information they receive. These messages represent emails, web forms, database entries, workflow data structures, and arguments to actions. We assume that messages have associated *tags* (e.g., "health information") to indicate their contents, but consider business processes in which human agents may tag messages incorrectly. This model forms a concurrent game structure [5] of agent actions that gives semantics to the logic of privacy and utility.

We also formulate the workflows themselves in temporal logic by associating a *responsibility* to each agent role. For example, in the patient portal workflow, doctors are responsible for answering health questions and secretaries are responsible for scheduling appointments. We consider both a general class of workflows presented abstractly by logical formulas and a more concrete subclass of practical workflows presented as a labeled graph or automata. Within this setting, we formulate and address the design-time and run-time questions below about whether a given workflow achieves its privacy and utility goals.

1. *Does a given workflow achieve privacy and utility if all agents act responsibly?* We present algorithms for answering this question, both by building on general results about LTL and by introducing the notion of a *local communication game*. Specifically, privacy properties may be evaluated using standard LTL model-checking over the traces generated by responsible executions of the concurrent game structure. Evaluating utility is more involved because of the ATL* path quantifiers and, in general, is undecidable [5] because agents learn only of messages they send or receive. Because of this limitation, we present a sound decision procedure for a restricted, but useful, class of formula defined.

2. *Can irresponsible agents be detected and held accountable for violations?*
   If the execution of a workflow satisfying the design criteria in question (1) above actually violates privacy, then some agent must have caused the violation by acting irresponsibly. These violations can be caught at run time, and the accountable agent determined using *auditing* algorithms we present. These algorithms are not fully automatic (or else they could be used for enforcement) but require an oracle (such as a human auditor) to determine the accuracy of message tags. We seek to minimize the number of oracle calls (reducing the human auditor's work) by using classical causality ideas in distributed computing and a new notion of "suspicious events."

Privacy advocates often recommend reconciling the competing interests of privacy and utility with the principle of *minimum necessary disclosure*: disclose the minimum information necessary to achieve the utility goal. This principle is included expressly in several influential privacy policies, including the HIPAA Privacy Rule and the Markle Connecting for Health Common Framework [49]. We leverage our unified model of privacy and utility to provide a formal definition of this principle.

We apply these concepts to the MyHealth patient portal and recommend several design changes to the MyHealth developers at Vanderbilt. Message tags are themselves one such suggestion, enabling finer grained message routing. Our auditing algorithms were developed in response to the MyHealth developers' concern about incorrectly tagged messages. We suggest further privacy improvements in the MyHealth workflow based on tagging and illustrate our auditing methods using a hypothetical execution of MyHealth with an irresponsible agent.

1.3 Related Work

There is a large body of related work. This section discusses related work from the literatures on access control, privacy, utility, and workflows. Chapter 2 details the anomalies in earlier privacy languages, and Section 4.2 compares our logic with other privacy languages.

1.3.1 Access Control

Discretionary. Unlike traditional access control policies (see, for example, [20, 25, 38]), which are designed to limit the actions of untrusted or malicious principals, privacy policies often are concerned with well-meaning, but perhaps non-expert, members of an organization. For example, health-care workers in a hospital are obligated to comply with HIPAA but might not be experts in what restrictions HIPAA places on their actions. A system that simply informed these individuals when they might be violating HIPAA would be useful in that they could more easily comply with the law. Moreover, health-care professionals are often leery of technology that prevents them from performing their duties. These professionals are skeptical of, and unwilling to adopt, an electronic medical record system that displays an "access denied" message to an emergency room doctor trying to save the life of a patient. For these reasons, we seek not to circumscribe actions, as one might in a digital rights management approach to privacy, but instead we seek to inform well-meaning principals about whether their actions comply with a privacy policy (and what future obligations those actions incur).

In comparison with access control and previous privacy policy frameworks, our norms focus on who personal information is about, how it is transmitted, and past and future actions by both the subject and the users of the information. Access control systems, conventionally, do not track whom information is about: permission to read or write a file might be granted or denied, but the decision is not based on who is described by the information in the file. Generally speaking, access control policies instruct a system as to whether a specific action is permitted, typically by deriving a relation between subjects, objects, and actions (possibly by grouping subjects by role). Conventional access control systems might make decisions based on the current state of the resources that it governs, but generally do not inquire about the past or impose restrictions on the future. In our model, the subject of information in a message is as important as the sender and the recipient of the message. For example, norms can permit doctors to communicate personal information about their patients but forbid them from communicating the personal information of their administrative

assistants. Additionally, temporal operators play a key role, capturing both "opt-in" (a past requirement) and confidentiality (a future requirement) using a single construct.

Mandatory. Another branch of the access control literature deals with mandatory access control [19]. These access control mechanisms often are design to enforce *non-interference*, preventing principals with "low" classification from learning information classified as "high." Although superficially similar to the information flow proscriptions in contextual integrity, these mandatory access control policies seek to prevent *all* information leakage whereas our model treats personal attributes as symbolic pieces of information. Consider, for example, the parking receipts from a hospital's parking garage. The time and date stamp on each receipt contains information about whether the owner of the car is on kidney dialysis. A strict non-interference policy would classify these receipts as "high" and prevent their disclosure to a principal classified "low." The logic of privacy and utility takes a more pragmatic position and does not automatically consider the receipts protected health information (as defined under HIPAA).

1.3.2 Privacy

P3P. The World Wide Web Consortium has proposed the Platform for Privacy Preferences (P3P) to enable service providers to post machine-readable privacy policies [1, 58, 27, 26, 21]. Built into Internet Explorer, P3P is the most widely deployed privacy language, but is also the least expressive. To make use of P3P policies, consumers (or consumer advocacy groups) express their privacy preferences in a language such as APPEL [27] or XPref [3]. The consumer's user agent then compares the consumer's privacy preferences with service provider's privacy promises. Service providers who post P3P policies promise specific data practices, but P3P itself is not designed to aid policy enforcement, although recent work [44] has examined enforcing P3P policies in database systems.

EPAL. The Enterprise Privacy Authorization Language (EPAL) [41, 11, 63, 13] is a privacy language designed by IBM to enforce privacy policies within the enterprise. EPAL, like the eXtensible Access Control Markup Language (XACML) [7, 6] on which it is based, defines the semantics of its policies in terms of an algorithm for evaluating policies. If Alice wishes to perform an action governed by a formal EPAL privacy policy, then she must first consult an authorization service. The authorization service, in turn, interprets the EPAL policy description and then responds with the set of restrictions that the policy levies on the desired action. Alice is then obligated to abide by those restrictions that the service returns.

Privacy APIs. Privacy APIs [50], is a privacy language based on a classic access control model. Privacy APIs are sufficiently expressive to capture the privacy provisions of HIPAA, although much of the complexity of data hierarchy and temporal conditions is relegated to a set of uninterpreted "evidence" strings. Other work [8, 9] formalizes privacy policies used by organizations but does not consider how to enforce those policies or how the policies affect the design of organizational workflow.

Group Privacy. In addition to work on individual privacy, a number of researchers have written papers about *group privacy*. In group privacy, also known as database privacy, a data aggregator, such as the Census Bureau or a hospital, releases sensitive attributes about a large number of individuals. The group privacy work focuses on disclosing as much aggregate information as possible while limiting what information an adversary can deduce about individual members of the group. Releasing too much information, even seemingly de-identified data, can often lead to the attacker identifying the individuals in the data [66, 51]. There are a number of techniques for anonymizing data. In k-anonymity [60] and l-diversity [47], the aggregator suppresses various database cells in the name of privacy. Other work suggests that the aggregator perform more substantial data transformations [33, 22] or limit which aggregate queries are available [42] to untrusted users. In contrast to the work on group privacy, we focus on individual privacy because our applications, health care providers and financial institutions, require access to individually identified personal information.

Some privacy regulations, including HIPAA, do provide for the disclosure of sanitized aggregate information about sensitive attributes, which is where techniques for group privacy find application.

1.3.3 Workflow and Utility

Several formalisms have been considered for specifying workflows, most notably Petri Nets [2], UML Activity Diagrams [32], and the Business Process Execution Language (BPEL) [53]. The Petri Net model is useful for understanding reachability and parallelism properties of workflows, but is difficult to apply to privacy because the stones in the net, which represent performing tasks or exchanging messages, are untyped (or, more precisely, typed implicitly by their location). The formalism for UML Activity Diagrams is graphical, complicating integration with the linguistic formalisms of privacy policies. Alternatively, BPEL, for which Oracle provides an execution engine [55], views a workflow as conglomeration of web services. Unfortunately, BPEL resembles an imperative programming language and is Turing-complete, foreclosing the possibility of deciding whether a BPEL workflow complies with a privacy policy or achieves a utility goal.

Workflow and authorization have been considered together previously, both for access control [10] and for privacy [67], but those works treat the workflow as given and do not consider the utility goal as a constraint on workflow design. Moreover, neither considers auditing deviations from prescribed workflow execution. Conflicts between privacy and utility goals have been recognized in other settings, such as in k-Anonymity [60, 66]: when a database cell is suppressed, privacy is enhanced and utility is diminished. When Dwork and Nissim [33] perturb a dataset to achieve privacy, utility concerns motivate them to minimize the amount of added noise.

1.4 Organization

The remainder of this thesis is organized as follows. Chapter 2 relates semantic anomalies in P3P and EPAL, both popular existing privacy languages, motivating

the improvements in our model. Chapter 3 defines the syntax and semantics of the logic of privacy and utility and details the fragment used to formalize contextual integrity. Chapter 4 evaluates the logic by developing its theory of combination, comparing its expressiveness with other privacy languages, and expressing three federal privacy regulations, HIPAA, COPPA, and GLBA. Chapter 5 expands the discussion to include utility and contains specific results about evaluating workflow design and auditing workflow logs. Chapter 6 studies the MyHealth@Vanderbilt patient portal in our model and recommends modifying the portal's workflow to improve its privacy protections. Finally, Chapter 7 concludes.

Chapter 2

Anomalous Privacy Languages

In this chapter, we examine semantic anomalies in two privacy languages, the Platform for Privacy Preferences (P3P) and the Enterprise Privacy Authorization Language (EPAL). These anomalies arise from the notion of a *data hierarchy*, which relates different attributes about a subject. Although related to the role hierarchy in role-based access control models, the data hierarchy presents a challenge unique to privacy languages. EPAL also has semantic anomalies in connection with *obligations*, which require a principal to discharge some duty after querying the privacy policy. Portions of this chapter appear in [17] and [16].

2.1 Platform for Privacy Preferences

A World Wide Web Consortium standard, the Platform for Privacy Preferences, or P3P, is a formal language for communicating privacy promises to consumers [27]. Many web sites deploy P3P policies [21], and the Internet Explorer web browser includes a P3P client. A P3P policy is a promise by a service provider to limit the use of particular data to particular purposes, recipients, and retention periods.

Prior to retrieving a web page, a consumer's web browser downloads the site's P3P policy and compares the policy against the consumer's privacy preferences. If the policy respects the user's preferences, the web browser retrieves the web page. However, if the policy does not respect the user's preferences, the browser may block

the site or notify the user. When manipulating data, the web site operator is obligated to adhere to the P3P policy under which it collected the data.

2.1.1 P3P Policies

P3P policies state privacy restrictions in terms of a data hierarchy, called the base data schema, or an extension thereof. A data hierarchy is a taxonomy of personal attributes, such as a subject's home telephone number or date of birth. These attributes are arranged in a hierarchy. For example, home telephone number is a sub-attribute of home contact information, which is a sub-attribute of contact information.

P3P policies are comprised of statements that attach a set of purposes, recipients, and retention periods to a set of personal attributes in the data hierarchy. In each statement, the policy's issuer promises to restrict its uses and disclosures of the specified attributes to the specified purposes, recipients, and retention periods. For example, in the P3P statement below, `webmd.com` promises to restrict use of the user's name and birthday to the "current" purpose and the "individual analysis" purpose.

```
<STATEMENT>
  <PURPOSE>
    <current/>
    <individual-analysis/>
  </PURPOSE>
  <RECIPIENT>
    <delivery required="opt-in"/>
  </RECIPIENT>
  <RETENTION>
    <legal-requirement/>
  </RETENTION>
  <DATA-GROUP>
    <DATA ref="#user.name" optional="yes"/><DATA ref="#user.bdate"/>
  </DATA-GROUP>
</STATEMENT>
```

Based on this policy, `webmd.com` may disclose the user's name and birthday to delivery agents, but this disclosure will be made only if the user opts in. Finally, `webmd.com` may retain the user's name and birthday to meet legal requirements on data retention.

A P3P statement attaching a promise to a node in the data hierarchy (for example, `#user.name`) also attaches that promise to all children of that node (for example, `#user.name.given`). If two statements attach different promises to the same data object, the issuer is permitted to perform actions permissible under either promise. Thus, requirements inherit down the hierarchy and a policy is the disjunction of its statements.

2.1.2 Perspectives on Privacy

Previous models of privacy policies [41] do not distinguish between the perspectives of those who handle personal data and those whose personal data is being handled. Such a distinction, however, is key to understanding the semantics of a privacy policy. Consider the following example. Alice is a consumer and Dr. Bob is a service provider. They are concerned with the Alice's blood cholesterol level, T-cell count, and blood test results. These attributes are related in the sense that both blood cholesterol level and T-cell count are types of blood test results (and hence descendants of blood test results in the data hierarchy). Alice and Bob are concerned with disclosure of these attributes.

Alice is HIV-positive. She wishes to keep her medical records private because she fears she will be denied health insurance if prospective insurers learn her HIV status. She could prohibit disclosure of her entire medical history, but she can obtain a better insurance rate if she permits some disclosures. She is willing to disclose some of her record, such as her age, weight, and x-rays, but she does not wish to disclose results of blood tests. In evaluating privacy policies, Alice decides to ask, "Does this policy permit disclosure of blood test results?"

After framing her question, Alice consults the privacy policy of her physician, Dr. Bob, to determine if he will respect her preference and keep her blood test results confidential. In his policy, Dr. Bob promises not to disclose blood cholesterol levels.

This is not sufficient to satisfy Alice because it does not preclude Dr. Bob from disclosing her T-cell count. Alice, therefore, concludes that Dr. Bob's policy does permit disclosure of some important blood test results.

Dr. Bob's perspective on his policy is different from Alice's perspective. In order to provide quality care for his patients, Dr. Bob wishes to disclose particular records, such as blood test results. In evaluating his privacy policy, he decides to ask, like Alice, "Does this policy permit disclosure of blood test results?" His policy promises not to disclose blood cholesterol levels, and therefore his policy prohibits him from disclosing blood test results in their entirety. He concludes his policy does not permit disclosure of arbitrary blood test results.

Alice and Dr. Bob appear to be asking the same question, but their different perspectives lead them to interpret their questions using different modalities. Alice is worried about Dr. Bob disclosing part of her blood test results, so she might restate her question as, "Does this policy permit disclosure of *some* blood test results?" Dr. Bob is worried about upholding his promises about blood test results, so he might restate his question as, "Does this policy permit disclosure of *all* blood test results?" The different modalities in these questions result in different answers for the same policy.

2.1.3 APPEL and XPref: Privacy Preferences

P3P policies are designed to be interpreted by automated user agents. A consumer's agent compares the data practices promised by a service provier's P3P policy with the consumer's privacy preferences, which are themselves expressed in a formal language. These privacy preferences let consumers specify which data practices they find acceptable and which they find unacceptable. If the user agent determines that the privacy policy does not conform to the consumer's preferences, the user agent can notify the consumer, letting the consumer make an informed decision about whether to use the service.

Semantically, a consumer's privacy preferences are a predicate over privacy policies, indicating whether the policy conforms to the preferences. One natural constraint on privacy preferences is that consumers prefer more restrictive policies. In

particular, if a consumer finds a particular policy acceptable, the consumer will also find all strictly more restrictive policies acceptable, and, conversely, if a consumer finds a particular policy unacceptable, the consumer will also find all strictly less restrictive policies unacceptable.

Formally, we say that consumer privacy preferences are *robust* if they are monotonic with respect the "strictness" ordering on privacy policies. We can view a robust set of privacy preferences as an upper bound on policy permissiveness. For example, a preference to block web sites that use home telephone numbers for telemarketing is robust, whereas a preference to block web sites that *do not* use home telephone numbers for telemarketing is not robust.

Non-robust preferences are somewhat nonsensical. Suppose a consumer wished to enforce a non-robust privacy preference that blocked access to web sites that did *not* use home telephone numbers for telemarketing. However, just because a web site reserves the right to use home telephone numbers for telemarketing does not mean they will actually use them. Although the consumer's privacy preference matches the site's policy, the consumer has not been promised that his or her home telephone number will be used for telemarketing. Ideally, a preference language would not be able to express non-robust preferences because a consumer who writes such a preference is almost certainly making an error.

APPEL. The P3P specification defines a language for expressing privacy preferences called A P3P Preference Exchange Language (APPEL) [27]. An APPEL preference reports whether interactions with a service provider espousing a P3P policy should be blocked, limited, or allowed to proceed. An APPEL preference is a set of rules, each of which consists of a judgment (`block`, `limited`, or `request`) and a condition under which to issue that judgment. Rules are processed in order, optionally halting at the first rule whose condition is met. A consumer can express non-robust privacy preferences in APPEL. For example, the following APPEL preference is not robust.

```
<appel:RULE behavior="block">
  <p3p:POLICY>
    <p3p:STATEMENT appel:connective="or">
      <p3p:PURPOSE appel:connective="non-and">
        <p3p:telemarketing />
      </p3p:PURPOSE>
    </p3p:STATEMENT>
  </p3p:POLICY>
</appel:RULE>
```

XPref. XPref is another language for expressing privacy preferences, based on XPath [23]. The designers of XPref were motivated by their difficulty expressing simple privacy preferences in APPEL [3]. Following from this motivation, the XPref designers endowed XPref with significant expressive power. Unfortunately, XPref can also express non-robust preferences. For example, the non-robust XPref rule below blocks services that do not use collected information for telemarketing.

```
<RULE behavior="block"
  condition="/POLICY/STATEMENT/PURPOSE/*
    [ name(.) != 'telemarketing' ]" />
```

Using XPref robustly is non-intuitive. For example, a number of the example XPref policies provided by the XPref designers in [3] are non-robust. Consider the privacy preference "block services that collect my home address." Naïvely, a consumer might expect the XPref rule below to express that preference:

```
<RULE behavior="block"
  condition="/POLICY/STATEMENT/DATA-GROUP/*
    [ name(.) = 'DATA' and @ref = '#user.home-info.postal' ]" />
```

However, this preference will not block a service whose privacy policy states that it collects `user.home-info` data (which includes postal address). A consumer's second attempt to encode this preference in XPref might be the rule below:

```
<RULE behavior="block"
  condition="/POLICY/STATEMENT/DATA-GROUP/*
    [ name(.) = 'DATA' and
      ( @ref = '#user.home-info.postal' or
        @ref = '#user.home-info' or
        @ref = '#user' ) ]" />
```

Unfortunately, this preference is still not accurate. A service provider might disclose that it collects each individual sub-element of `user.home-info.postal`, such as `street`, without explicitly disclosing that it collects home addresses. The XPref rule below accurately expresses this preference by correctly capturing the consumer's *some* modality:

```
<RULE behavior="block"
  condition="/POLICY/STATEMENT/DATA-GROUP/*
    [ name(.) = 'DATA' and
      ( starts-with(@ref, '#user.home-info.postal') or
        @ref = '#user.home-info' or
        @ref = '#user' ) ]" />
```

2.2 Enterprise Privacy Authorization Language

The Enterprise Privacy Authorization Language (EPAL) [63], promoted by IBM [41], is a formal language for expressing privacy policy. By formalizing its privacy policy in EPAL, an organization can mechanically enforce its privacy privacy policy. Typically, an EPAL policy is evaluated at a *policy decision point*, which is queried by a *policy enforcement point* responsible for enforcing the privacy policy.

Given a contemplated action, an EPAL policy evaluates to either an allow, a deny, or a "don't care" judgment. The "don't care" judgment means that this policy neither allows nor denies the action. In addition to allowing or denying actions, EPAL policies can entail *obligations* that require or prohibit future actions. For example, an airline company's privacy policy might permit a hotel affiliate to learn a frequent

traveller's email address but then require the affiliate to notify the traveller and let him or her opt out of future promotions.

Similar to data attributes, the various obligations in an EPAL policy might be related to each other. For example, the obligation to expunge a travel itinerary within one week subsumes an obligation to expunge the itinerary within one month in the sense that complying with the first necessarily entails complying with the second. Two obligations might also be incompatible. For example, the obligation to expunge an itinerary within one month is incompatible with the obligation to retain the same itinerary for one year in the sense that it is impossible to fulfill both obligations.

2.2.1 Desiderata

To evaluate EPAL, we propose four properties a well-designed privacy language could hope to achieve. Ideally, a privacy policy language should force *consistency*, permit *local reasoning*, guarantee *safety*, and allow policy *combination*.

Consistency. A privacy policy that both permits and forbids a specific action is *inconsistent*. More subtly, a policy that entails two incompatible obligations for the same action is also inconsistent. For example, a privacy policy that, after performing a specific action, entails both the requirement to "expunge within one week" and the requirement to "retain for one year" is inconsistent because both obligations cannot be fulfilled simultaneously.

In some sense, inconsistent policies can still be useful. For example, suppose a policy enforcement point queries the policy and determines that a particular action is both allowed and denied (or that the action entails incompatible obligations). The policy enforcement point can prevent the action from occurring, thereby avoiding the contradiction. However, this situation requires the policy enforcement point to make a policy decision and prevents the policy itself from arbitrating which actions are and are not permitted.

Local Reasoning. Suppose an airline's privacy policy explicitly states that affiliated hotels must check an opt-out list before using email addresses obtained from

the airline. A policy auditor might wish to conclude that the entire policy actually enforces this statement. Some privacy languages, however, do not make this a valid conclusion. Policy languages that admit such *local reasoning* are easier to maintain because policy authors need only consider local portions of the policy when deciding whether a complex policy accurately reflects their intent.

Intuitively, a policy language admits local reasoning if every policy written in the language enforces each of its statements, letting a policy author draw conclusions about the entire policy from only a fragment of the policy. This property is a bit challenging to formalize, but one approach is to endow each policy statement with semantics independent from the policy in which the statement appears. A policy language, then, admits local reasoning if, for each policy in the language, these locally computed semantics are implied by the global semantics of the entire policy.

Safety. Privacy policies can be checked for internal consistency. For example, the requirements on disclosing a subject's home address must be at least as strict as the requirements on disclosing the subject's entire home contact information (because home contact information contains home address). Imposing fewer restrictions on disclosing home contact information than on disclosing home addresses is *unsafe* because a member of the enterprise can mistakenly violate the policy by disclosing a subject's entire home contact information and thereby disclose the subject's home address.

In order for a policy to be safe, the entailed obligations must also be internally consistent. For example, consider a policy with two requirements: one that permits a computer system to read a subject's home address provided the system expunges the record within one month and another that permits the computer system to read a subject's home contact information provided the system expunges the record within one year. This policy also is unsafe because the computer system incurs a weaker obligation (expunge within one year) for a less specific action (read a travel history). The policy would be safe, however, if the policy required the system to expunge the travel history within one *week* because the less specific action entails a stronger obligation.

Combination. Often, enterprises wish to combine privacy policies. For example, an enterprise might aim to abide both by privacy regulations and by an internal policy. By combining its own internal policy with a standard industry-wide policy written by a policy provider, the enterprise can ensure compliance with both policies. Partnering or other outsourcing agreements also motivate policy combination, for example, as in the JetBlue case study [9]. A privacy language is *closed under combination* if the language can express the conjunction of any two policies expressible in the language.

Conjunction is but one policy combination operator. An enterprise might wish to take the disjunction of two policies, for example, to determine which disclosures are possible if one department implements one privacy policy and another department implements another privacy policy. More precisely, a privacy language is closed under a given policy combination operator (such as conjunction or disjunction) if the language can express the result of combining any policies expressible in the language.

2.2.2 Evaluation

Consistency. Each EPAL policy rule contains a condition and applies only to queries satisfying the condition. Given a query, the authorization service examines policy rules in order and collects obligations from each applicable rule. The service stops examining rules once it encounters an applicable statement containing an "allow" or "deny" judgement. This algorithm causes an EPAL rule to be enforced only if (1) the query matches the rule's condition and (2) no earlier applicable rule contains an "allow" or "deny" judgment. These sequential semantics force consistency because the authorization service terminates policy prior to reaching conflicting statements.

An EPAL policy can be inconsistent if the authorization service collects incompatible obligations before reaching an "allow" or "deny" judgment. However, the designers of EPAL assume all obligations are compatible [41]. Although privacy languages that force consistency might seem appealing, forcing consistency has some disadvantages. For example, an attempt both to permit and to prohibit a situation might be viewed as a mistake by the policy author. By forcing consistency, EPAL hides such mistakes.

Local Reasoning. EPAL's sequential semantics render local reasoning unsound. Because earlier policy rules can shadow later rules, a policy auditor must consider all preceding rules in order to understand a given rule. Worse, a wholly unreachable rule has no effect on the meaning of the policy and could mislead policy authors. For example, the EPAL policy below does not actually require hotels to check the opt-out list.

```
<epal-policy>
  <rule id="a" ruling="allow">
    <user-category refid="Business affiliates"/>
    <data-category refid="Email addresses"/>
  </rule>
  <rule id="b">
    <user-category refid="Affiliated hotels"/>
    <data-category refid="Email addresses"/>
    <obligation    refid="Check opt-out list"/>
  </rule>
</epal-policy>
```

If an affiliated hotel queries the policy regarding use of travellers' email addresses, the authorization service will stop evaluating the policy once it processes rule `a` and will not encounter rule `b`, which requires consulting the opt-out list. If these policy statements were contained in a larger policy, perhaps separated by pages of rules, an auditor might incorrectly conclude that affiliated hotels are required to check the opt-out list.

Safety. EPAL can express unsafe policies. To see this, consider a policy author who tries to fix the above policy by interchanging rule `a` and rule `b`. The interchanged policy is unsafe because an affiliated hotel can avoid being required to check the opt-out list by querying the authorization service as a generic business affiliate. EPAL can express unsafe policies because EPAL does not require that more general queries lead to more stringent obligations. Unsafe policies are problematic because they

let members of the enterprise avoid obligations, either accidentally or maliciously, undermining the intent of the policy's author.

Combination. EPAL is not closed under combination. For example, a policy that allows an action cannot be combined with a policy that denies an action because the resultant policy would be inconsistent and inconsistent policies cannot be expressed in EPAL. Moreover, two compatible EPAL policies (whose combination would be consistent) might have a combination that is expressible in EPAL. In particular, obligations attached to "deny" judgments can preclude combination. For example, consider the following two EPAL policies:

1. If a member of the marketing department attempts to access a traveller's passport number, then the access is denied and must be logged in marketing's privacy log.

2. If a member of the human resources department attempts to access a traveller's passport number, then the access is denied and must be logged in human resources' privacy log.

In the combined policy, if a member of an undetermined department attempts to access a traveller's passport number, then the access is denied and must be logged in both marketing's log and human resources' log. Although policies are compatible and their combination is consistent, the combined policy cannot be expressed in EPAL. (There exists an extension of EPAL that is closed under combination [12].)

2.2.3 Perspective

Ideally, a privacy policy language should force consistency, permit local reasoning, guarantee safety, and allow policy combination. However, after defining these properties precisely, we observe that these four goals cannot be achieved simultaneously in a language that provides a minimum level of expressiveness. Although EPAL is designed to support consistency at the cost of local reasoning and combination, we believe that it is possible to make better design trade offs. In particular, because

consistency of an individual policy can be determined by a practical algorithm, we believe it is better to use a privacy language that supports local reasoning and policy combination and to rely on tools for assuring consistency.

Chapter 3

A Logic of Privacy and Utility

In this chapter, we develop a logic for reasoning about privacy and utility. The logic is motivated by the philosophical theory of contextual integrity [52]. The syntax is based on Alternating-time Temporal Logic (ATL*) [5], and the semantics are based on a specific concurrent games structure of communicating agents. This systematic development of the language avoids the anomalous semantics that have plagued other privacy languages. Portions of this chapter appear in [14] and [15].

3.1 Overview of Contextual Integrity

Contextual integrity is a philosophical account of privacy in terms of the transfer of personal information. It is not proposed as a full definition of privacy, but as a normative model, or framework, for evaluating the flow of information between agents (individuals and other entities), with a particular emphasis on explaining why some patterns of flow provoke public outcry in the name of privacy (and why some do not). In the approach encompassed by contextual integrity, the intricate systems of social rules governing information flow are the crucial starting place for understanding normative commitments to privacy. Although *contextual integrity* is itself a relatively recent term, the idea of contextually relative norms has been "in the air," recognized in various ways in the literature (e.g., [57, 61, 62]), and explored in some specific ways in a variety of work dealing with professional confidentiality rules. Four constructs

are key to defining contextual integrity: informational norms, appropriateness, roles, and principles of transmission.

We begin, however, with the concept of a *context* to capture the idea that people act and transact in society not simply as individuals in an undifferentiated social world, but as individuals in particular capacities (roles), in distinctive social contexts, such as health care, education, employment, the marketplace, and so on. These contexts should be understood as structured settings whose features have evolved over time—sometimes long periods of time—subject to a host of contingencies of place, culture, historical events, and more. Features that characterize particular contexts include the assemblages of roles (sometimes open-ended) and the set of behavior-guiding norms that prescribe (and proscribe) actions and practices, when, for example, people consult a physician (or are the physician), attend school (or teach), and shop (or sell).

One further feature is key to understanding what we mean here by "contexts," for they are characterized not only by roles and norms but also by particular ends, or values. In the case of health care, an onlooker (say, from another planet) observing a typical health care setting of a hospital, will be unable make proper sense of the goings-on without appreciating the underlying purpose behind it, that is, alleviating illness and promoting health. Although settling the exact nature of the ends and values for any given context is not a simple matter—even in the case of health care, which is relatively robust—the central point is that the roles and norms of a context make sense, largely, in relation to those values. Because this point, although relevant to the larger theory of contextual integrity, is not crucial to the specific goals of this paper, we will not elaborate on it any further. Instead, our formalization deals with contexts frozen at a particular moment in history, focusing on expressing their attendant norms precisely.

For purposes of understanding privacy, norms that apply to the transmission (or communication) of personal information from one party to another, which we call "informational norms," are singularly important. In a health care context, for example, informational norms limit what physicians can say to others about the health

condition of patients under their care. Contextual integrity, then, is a feature of situations in which the informational norms of a context have been respected; when any of these norms have been unjustly breached, then we say that contextual integrity has been violated.

One of the key defining aspects of informational norms, and judgments that contextual integrity has or has not been violated, is the type (category, nature, class) of information in question. Unlike a number of prominent normative accounts of privacy, the approach taken here rejects the idea that a simple dichotomy—usually between public and private (sensitive, intimate) information—is sufficient for adjudicating privacy claims. Instead, there is potentially an indefinite variety of types of information that could feature in the informational norms of a given context. We suggest the term "appropriateness" as a way to signal whether the type of information in question conforms to the relevant informational norms. Thus, for example, in the context of a job interview for the position of bank manager in the present-day United States, information about applicants' marital status is inappropriate, but it is appropriate in the context of dating (or courtship). Because information type is so salient an influence on people's judgments that a violation has occurred, earlier accounts of contextual integrity had posited norms of appropriateness as distinct from norms of transmission. Our effort to formalize contextual integrity reveals, however, that, at some level of generality, both can be covered by the form of transmission norm explored in this paper.

Associated with every communication there are three relevant entities (agents, principals): the one from whom the information flows, the one to whom the information flows, and the one—the information subject—about whom the information is. Entities are considered to be acting in particular capacities, or roles, which are articulated with varying degrees of detail, within the relevant contexts. In academic departments, for example, the roles of chair, tenured faculty, assistant professor, student, administrator, and so forth, each are associated with a set of duties and privileges. Thus, contextual integrity maintains that roles are key variables affecting the rich and complex sensibility people demonstrate in their judgments over whether a violation has occurred.

The notion of a transmission principle may be the most distinctive aspect of the approach to privacy through contextual integrity. These principles are the specific constraints (terms or conditions) regulating flow of information from entity to entity prescribed by informational norms. One such principle is confidentiality, prohibiting agents receiving information from sharing it with others in the future. Although confidentiality is prominent, there are many other principles of transmission, for example, reciprocity, determining that information flow is bi-directional (occurring in friendship but not between a patient and a physician). Another is dessert, determining that an agent deserves to know or learn something about the subject, perhaps, people deserving to know whether their lovers are HIV positive. An important family of transmission principles hinges on the awareness and consent of the information subject; in one instance, a subject might be forced to reveal information, in another, a subject might know (or not know) whether information has been transmitted, in a third, the subject consents to transmit information, and so on. Norms prescribe which transmission principles ought to govern the flow of information and are understood to be violated if the principles are not followed. It is worth noting that control by subjects of the flow of information about themselves, which features definitively in some theories, is merely one transmission principle—albeit an important one—among many. There is probably no end to the variation in transmission principles.

3.2 Model

In this section, we formalize a fragment of contextual integrity. Our model consists of communicating agents who take on various roles in contexts and send each other messages containing attributes of other agents. The evolution of the knowledge of individual agents depends on messages they receive and computation rules that enable agents to infer further attributes. We begin with a model of communicating agents in which agents send each other messages containing personal information about each other. Upon receipt of a message, an agent incorporates the contents of the message into his or her knowledge state, enlarging the set of possible messages the agent can send in the future.

3.2.1 Attributes, Agents, and Messages

Attributes. Let $\mathcal{T}$ be a set of *attributes*. Some attributes are related to other attributes. For example, Alice can compute Bob's *postal-code* given Bob's *mailing-address*. We model this by equiping $\mathcal{T}$ with a partial order $\preceq$. Intuitively, if Alice knows the value of attribute t for Bob, then Alice can compute the value of each attribute $t' \preceq t$ for Bob. Thus, *postal-code* $\preceq$ *mailing-address*. We omit "group" attributes, for example the average height of Alice, Bob, and Carol.

Agents. Let $\mathcal{P}$ be a set of *agents*. Agents themselves lack structure, but are associated with a set of attributes, the attribute about that agent, and interact by sending each other messages. Let $\mathcal{M}$ be a set of *messages*. Associated with each message $m \in \mathcal{M}$ is a set contents$(m) \subseteq \mathcal{P} \times \mathcal{T}$, indicating what attributes about which agents are actually contained in the message. The contents of each message is downwardly closed under $\preceq$. Formally, if $t' \preceq t \in$ contents(m), then $t' \in$ contents(m).

For example, Alice, Bob, and Carol are agents. Alice can send Bob a message containing Carol's postal address. Because the contents of messages are closed under $\preceq$, the message necessarily also contains Carol's postal (zip) code. Intuitively, when Bob receives Carol's postal address, he can determine Carol's postal code, and the world is as if the postal code were included explicitly in the text of the message. This direct inclusion of the data hierarchy relation, $\preceq$, in the model helps avoid semantic anomalies arising from the data hierarchy.

For some applications, we wish to distinguish message features that are readily apparent to machines from those that require semantic understanding to observe. To model this effect, we associate a set of tags$(m) \subseteq \mathcal{P} \times \mathcal{T}$ with each message m, indicating the *tags* carried by the message. The tags of a message are the purported contents of the message and need not bear any relation to the actual contents of the message. Human agents can ascertain the actual contents of the message, whereas mechanical agents have access only to the tags.

Actions. Both $\mathsf{Send}(p, q, m)$ and $\mathsf{Receive}(p, q, m)$ are *actions* where $p, q \in \mathcal{P}$ are agents and $m \in \mathcal{M}$ is a message. The action $\mathsf{Send}(p, q, m)$ occurs when agent p sends

message m to agent q, and the action $\mathsf{Receive}(p, q, m)$ occurs when agent q receives message m from agent p. We focus on synchronous communication for which the Send and $\mathsf{Receive}$ actions occur simultaneously, but we include both actions for generality.

3.2.2 Knowledge, Communication, and Moves

Knowledge. Associated with each agent is a collection of the attributes that agent knows. A *knowledge state* κ is a subset of $\mathcal{P} \times \mathcal{P} \times \mathcal{T}$. If $(p, q, t) \in \kappa$, we say agent p *knows* the value of attribute t of agent q. For example, Alice knows Bob's height. The attribute relation $\preceq$ induces a transition relation on knowledge states. We write $\kappa \xrightarrow{\preceq} \kappa'$ whenever:

1. $\kappa' = \kappa$ or
2. $\kappa' = \kappa \cup \{(p, q, t')\}$, where $t' \preceq t$ and $(p, q, t) \in \kappa$, or
3. There exists κ'' such that $\kappa \xrightarrow{\preceq} \kappa''$ and $\kappa'' \xrightarrow{\preceq} \kappa'$.

Intuitively, $\kappa \xrightarrow{\preceq} \kappa'$ if a set of agents in knowledge state κ can arrive at knowledge state κ' by reasoning only individually, i.e. without communicating with each other.

Communication. In order to send a message m, the sending agent must know the contents of m. After receiving m, the receiving agent learns the contents of m. For example, Alice can send a message to Bob containing Carol's height just in case Alice herself knows Carol's height. After receiving such a message, Bob learns Carol's height. Associated with each knowledge state is a set of actions *available* to p:

$$\text{available}_p(\kappa) = \{\tau\} \cup \{\mathsf{Send}(p, q, m) \mid \{p\} \times \text{contents}(m) \subseteq \kappa\},$$

where τ is the null action that does not transmit a message. Notice that agents are free to select arbitrary tags when sending messages. Recipients learn the contents of messages they receive: $\kappa \xrightarrow{\mathsf{Send}(p,q,m)} \kappa'$ where κ' is a knowledge state such that

$$\kappa \cup \{q\} \times \text{contents}(m) \xrightarrow{\preceq} \kappa'.$$

The relation $\xrightarrow{\tau}$ is the identity relation on knowledge states. For every finite set of actions $A = \{a_1, \ldots, a_n\}$ available in κ, let $\kappa \xrightarrow{A} \kappa'$ if $\kappa \xrightarrow{a_1} \cdots \xrightarrow{a_n} \kappa'$. This is well-defined because available_p is monotonic on $\rightarrow$ and the resulting knowledge state is independent of the enumeration of A.

Moves. We add operational semantics by embedding the labeled transition system of knowledge states into a concurrent game structure [5], which we refer to as $\mathcal{G}$. If $\mathcal{G}$ is currently in state κ, each agent selects a *move*, a set of actions $A_p \subseteq \text{available}_p(\kappa)$, and $\mathcal{G}$ advances to the unique maximal knowledge state κ' such that

$$\kappa \xrightarrow{\bigcup_p A_p} \kappa'.$$

An action $\mathsf{Send}(p_1, p_2, m)$ is *visible* to agent p if p is the sender, p_1, or the recipient, p_2. An agent p's *view* of a trace π is the subsequence $\pi \restriction p$ containing all and only those actions visible to p. Agents are unaware of actions outside their view. Each agent p decides which moves to make according to a *local strategy*, a function from finite p-views to moves for p. This locality requirement makes $\mathcal{G}$ a game of imperfect information.

3.2.3 Roles and Contexts

Following contextual integrity, we associate agents with roles as part of contexts. Let $\mathcal{R}$ be a set of *roles* and $\mathcal{C}$ be a partition of $\mathcal{R}$. We refer to elements $c \in \mathcal{C}$ as *contexts* and the roles $r \in c$ as the roles of context c. For example, "teller" is a role in a banking context and "doctor" is a role in a health care context. The roles are structured by a partial order $\leq_{\mathcal{R}}$. If $r_1 \leq_{\mathcal{R}} r_2$, then r_1 is a specialization of role r_2 and, symmetrically, r_2 is a generalization of r_1. For example, a psychiatrist is a specialization of a doctor, which in turn is a specialization of a health care provider.

Agents can be active in multiple roles simultaneously. For example, Alice can be at once a doctor in a health care context and a customer in a banking context. A *role assignment* ρ is a subset of $\mathcal{P} \times \mathcal{R}$. If $(p, r) \in \rho$, we say agent p is active in, or plays, role r. For example, if $(\text{Alice}, \textit{psychiatrist}) \in \rho$, then Alice is active in the role

of psychiatrist. We require role assignments to be closed under role generalization, that is if $r_1 \leq_{\mathcal{R}} r_2$ and $(p, r_1) \in \rho$, then $(p, r_2) \in \rho$. Returning to our example, if $(\text{Alice}, \textit{psychiatrist}) \in \rho$, Alice must be active in the role of doctor in addition to that of psychiatrist. There are many instances of each context (many banks, many hospitals), but for clarity we omit instances.

3.3 Logic

We employ a standard logical notation for expressing privacy policies. This facilitates the technical development and clearly connects the language to the semantics based on communicating agents. We first present the syntax of the logic first and the imbue the syntax with semantics.

3.3.1 Syntax

The syntax of the logic is a particular signature of Alternating-time Temporal Logic (ATL*) [5]. For example, if Alice tells Bob her age under the principle of confidentiality, then, in the future, Bob must not disclose Alice's age. The past operators are also useful for capturing "opt-in" and other similar privacy idioms. We use the fragment of the logic in Linear-time Temporal Logic [48] to capture Contextual Integrity's norms of transmission. We include the ATL strategy quantifier $\langle\langle \vec{p} \rangle\rangle \varphi$ to capture notions of utility. We employ both the $\mathsf{contains}$ and the tagged predicates to be able to compare the purported and actual contents of messages.

$$\begin{aligned} \varphi ::= {} & \mathsf{send}(p, q, m) \mid \mathsf{inrole}(p, r) \mid \mathsf{incontext}(p, c) \mid \\ & t_1 \preceq t_2 \mid \mathsf{contains}(m, p, t) \mid \mathsf{tagged}(m, p, t) \mid \\ & \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \mathbf{U} \varphi \mid \varphi \mathbf{S} \varphi \mid \mathbf{X}\varphi \mid \langle\langle \vec{p} \rangle\rangle \varphi \mid \exists x.\varphi \end{aligned}$$

Informally, the predicate $\mathsf{send}(p, q, m)$ indicates that agent p has just sent agent q a message m; $\mathsf{inrole}(p, r)$ indicates that agent p is in role r; $\mathsf{incontext}(p, c)$ indicates that agent p belongs to a role in context c; $\mathsf{contains}(m, p, t)$ indicates that message

m contains attribute t about agent p; $\mathsf{tagged}(m, p, t)$ indicates that the message m is tagged with tag t about agent p; $t_1 \preceq t_2$ indicates that attribute t_1 can be computed from attribute t_2. $\varphi \mathbf{U} \psi$ holds if, and only if, φ holds until ψ holds (ψ must eventually hold). The modality "since," written $\mathbf{S}$ is the past version of $\mathbf{U}$. $\mathbf{X}\varphi$ holds if, and only if, φ holds in the next state. Finally, $\exists$ is rigid existential quantification.

We use the standard abbreviations $\wedge$, $\rightarrow$, and $\forall$ as usual in first order logic and $\mathbf{F}\varphi \equiv \mathit{true}\mathbf{U}\varphi$ for "eventually" and $\mathbf{G}\varphi \equiv \neg\mathbf{F}\neg\varphi$ for "henceforth" as is usual in temporal logic (e.g., [48]). A formula φ is an *LTL formula* if it is free of strategy quantifiers.

3.3.2 Semantics

Formulas are interpreted over the concurrent game structure $\mathcal{G}$ as usual for ATL [5] with imperfect information. A formula $\langle\langle \vec{p} \rangle\rangle \varphi$ holds if, and only if, a set of agents $\vec{p}$ has a local strategy to bring about φ. That is there exists a function mapping $\vec{p}$'s view of the history thus far to moves for each agent in $\vec{p}$ that ensures φ holds regardless of the actions of other agents.

An *environment* is a function η from variables to $\mathcal{P} \cup \mathcal{T} \cup \mathcal{M} \cup \mathcal{R} \cup \mathcal{C}$. For a variable x, we write $[\![x]\!]\eta = \eta(x)$. For technical convenience, we record the most recently performed actions along with the knowledge state κ as the state of $\mathcal{G}$. In a given state (κ, A) (where κ is a knowledge state and A is a set of actions), a role assignment ρ, and an environment η, the predicates are defined as follows:

$$
\begin{aligned}
&\kappa, A, \rho, \eta \models \mathsf{send}(p_1, p_2, m) \\
&\qquad \iff \mathsf{Send}([\![p_1]\!]\eta, [\![p_2]\!]\eta, [\![m]\!]\eta) \in A \\
&\kappa, A, \rho, \eta \models \mathsf{inrole}(p, r) \\
&\qquad \iff ([\![p]\!]\eta, [\![r]\!]\eta) \in \rho \\
&\kappa, A, \rho, \eta \models \mathsf{incontext}(p, c) \\
&\qquad \iff \text{There exists } r \in [\![c]\!]\eta \text{ such that } ([\![p]\!]\eta, r) \in \rho \\
&\kappa, A, \rho, \eta \models t \preceq t' \\
&\qquad \iff [\![t]\!]\eta \preceq [\![t']\!]\eta \\
&\kappa, A, \rho, \eta \models \mathsf{contains}(m, q, t) \\
&\qquad \iff ([\![q]\!]\eta, [\![t]\!]\eta) \in \mathrm{content}([\![m]\!]\eta) \\
&\kappa, A, \rho, \eta \models \mathsf{tagged}(m, q, t) \\
&\qquad \iff ([\![q]\!]\eta, [\![t]\!]\eta) \in \mathrm{tags}([\![m]\!]\eta) \\
&\kappa, A, \rho, \eta \models \varphi_1 \wedge \varphi_2 \\
&\qquad \iff \kappa, A, \rho, \eta \models \varphi_1 \text{ and } \kappa, A, \rho, \eta \models \varphi_2 \\
&\kappa, A, \rho, \eta \models \neg\varphi \\
&\qquad \iff \kappa, A, \rho, \eta \not\models \varphi
\end{aligned}
$$

The temporal operators and the strategy quantifier are defined as usual [5] for ATL*. For the temporal modality **S**, we choose the linear past semantics. To simplify notation, we use the following standard symbols:

$$
\begin{aligned}
\varphi_1 \vee \varphi_2 &\equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2) & \qquad \varphi_1 \rightarrow \varphi_2 &\equiv \neg\varphi_1 \vee \varphi_2 \\
\mathbf{F}\varphi &\equiv \top \mathbf{U} \varphi & \qquad \mathbf{G}\varphi &\equiv \neg\mathbf{F}\neg\varphi \\
\mathbf{F}^{-}\varphi &\equiv \top \mathbf{S} \varphi & \qquad \mathbf{G}^{-}\varphi &\equiv \neg\mathbf{F}^{-}\neg\varphi \\
\varphi_1 \mathbf{W} \varphi_2 &\equiv \varphi_1 \mathbf{U} \varphi_2 \vee \mathbf{G}\varphi_1 & \qquad \varphi_1 \mathbf{B} \varphi_2 &\equiv \varphi_1 \mathbf{S} \varphi_2 \vee \mathbf{G}^{-}\varphi_1 \\
\forall x.\varphi &\equiv \neg\exists x.\neg\varphi
\end{aligned}
$$

The formula $\mathbf{F}\varphi$ is read "eventually φ," and indicates that φ will eventually hold. Its dual modality, $\mathbf{G}$, is read "henceforth." The modalities $\mathbf{F}^-$ and $\mathbf{G}^-$ are the past forms of $\mathbf{F}$ and $\mathbf{G}$, respectively. We will often write $\sigma \models \varphi$ in place of $\sigma, 0, \eta \models \varphi$ when φ has no free variables (and thus does not depend on η).

3.3.3 Extensions

There are number of possible extensions to the logic. Typically, the extensions provide added expressiveness at a small complexity cost. Here, we discuss two such extensions.

Actions. Formally, the syntax of the logic contains only send actions. Other actions can be faithfully represented as sending messages, as in object-oriented programming languages such as Smalltalk and Java. The formula below expresses that a customer's telephone number should not be used for telemarketing:

$$\mathbf{G}\forall p, q, d.\, \mathsf{inrole}(p, \mathit{customer}) \wedge \mathsf{contains}(d, p, \mathit{telephone\text{-}number}) \rightarrow \neg\, \mathsf{send}(q, \mathit{telemarketing}, d)$$

Syntactic sugar could be added to more naturally express general actions without affecting the technical results of this thesis provided all the actions are recorded in the audit log.

Parameterized Roles. Some policies require finer distinctions between roles. For example, in order for Alice to perform an action, the policy might require that not only that is Alice a doctor, but also that she is Bob's doctor. The logic can be extended to express these parameterized roles by replacing the two-ary inrole predicate with a three-ary inrole predicate, as in $\mathsf{inrole}(\mathit{Alice}, \mathit{Bob}, \mathit{doctor})$. Semantically, roles become pairs containing a role identifier and an agent. Again, the main results of the thesis are unaffected.

Purpose. A number of policies depend on the *purpose*, or motivating reason, behind a communication action. (Note that this use of the term differs from the use of the

term in Contextual Integrity.) For example, some policies contain provisions that permit some communications for shipping purposes but not for other purposes. Often these provisions can be expressed by restricting the roles of message recipients (for example to delivery agents), but we can also extend the language to contain purpose directly by adding another parameter to the send predicate.

3.4 Formalization of Contextual Integrity

Contextual Integrity's norms of transmissions are expressible in an LTL fragment of the logic of privacy and utility. Each norm is either positive or negative. A positive norm might state that doctor Alice can send patient Carol's test results to researcher Bob *if* Bob keeps the records in confidence. Negative norms are dual: they state communication can occur only if the temporal condition is satisfied. For example, doctor Alice can send patient Carol's test results to researcher Bob *only if* Bob keeps the records in confidence. In the positive case, some other norm could authorize the communication and Bob would not be obliged to keep the results confidential, whereas in the negative case Bob must keep the results confidential regardless of how he obtained them from Alice.

We say a trace σ *satisfies the norms of context* c if the formula below holds. The formula below takes a disjunction over the *positive norms of transmission* for context c, denoted $\text{norms}^+(c)$, and a conjunction over the *negative norms of transmission* for context c, denoted $\text{norms}^-(c)$. Thus, in order to satisfy the norms, a communication must be allowed by at least one of the positive norms and it must respect all of the negative norms.

$$\sigma \models \mathbf{G}\forall p_1, p_2, q.\forall m.\forall t.$$
$$\mathsf{incontext}(p_1, c) \wedge \mathsf{send}(p_1, p_2, m) \wedge \mathsf{contains}(m, q, t) \rightarrow \bigvee_{\varphi^+ \in \text{norms}^+(c)} \varphi^+ \wedge \bigwedge_{\varphi^- \in \text{norms}^-(c)} \varphi^- \quad (3.1)$$

$$\text{positive norm:} \quad \mathsf{inrole}(p_1, \hat{r}_1) \wedge \mathsf{inrole}(p_2, \hat{r}_2) \wedge \mathsf{inrole}(q, \hat{r}) \wedge (t \in \hat{t}) \wedge \theta \wedge \psi$$

$$\text{negative norm:} \quad \mathsf{inrole}(p_1, \hat{r}_1) \wedge \mathsf{inrole}(p_2, \hat{r}_2) \wedge \mathsf{inrole}(q, \hat{r}) \wedge (t \in \hat{t}) \wedge \theta \rightarrow \psi$$

The syntactic forms of positive and negative norms are depicted above, where θ is an *agent constraint* and ψ is a *temporal condition*. An agent constraint θ is a formula free of temporal operators with free variables among p_1, p_2, and q. It expresses a relation among the sender, the recipient, and the subject, for example, that the sender and the subject are one and the same agent. A temporal condition ψ formalizes the notion of a principle of transmission and is a temporal formula with free variables among p_1, p_2, q, m, and t. It requires particular future actions to occur and particular past actions to have occurred (see Section 4.3 for concrete examples of norms).

One subtle consequence of the construction of Formula (3.1) is the treatment of attributes. Each individual norm applies to a downwardly closed set of attributes (downward in the information ordering on attributes induced by the computation rules). This captures the usual implication that the statement "allow disclosure of postal address" also allows the disclosure of postal codes. The formula universally quantifies over attributes because each communicated attribute must have a normative basis. The usual "upwards" inheritance of deny rules arises naturally here from the universal quantification over attributes and the downward closure of message contents. Suppose, for example, a norm denies the disclosure of postal codes. If one agent attempts to send a message containing a postal address, that message must also contain a postal code and when the attribute "postal code" is considered by the universal quantifier, the formula will forbid the disclosure.

Chapter 4

Evaluation of the Logic

In this chapter, we evaluate the Logic of Privacy and Utility (LPU) from the preceding chapter along three dimensions. First, we examine the algorithmic properties of consistency, entailment, combination, and compliance. Second, we compare the expressiveness of LPU with existing access control and privacy languages. Third, we express the main privacy provisions of three privacy regulations, HIPAA [54], COPPA [35], and GLBA [36], in the language. Portions of this chapter appear in [14].

4.1 Policies, Combination, and Compliance

A privacy policy regulates what flows of information are permitted between agents in various roles. A policy is a conjunction of contexts, requiring the norms of each context to be respected. For example, if Alice plays roles in both a bank and a hospital, she must act in accordance with the informational norms of both contexts.

Def. A *privacy policy* is a conjunction of formulas of the form (3.1).

We define below methods for evaluating privacy policies, both independently and in comparison with other policies. In addition, we define a notion of privacy compliance for an action. These problems can be solved using standard tools because they are formulated in LTL.

4.1.1 Consistency

A policy is consistent if it is possible for communicating agents to respect the policy. Inconsistent policies are not useful because they prescribe norms that agents cannot possibly satisfy. As defined, privacy policies can be satisfied trivially by agents who refrain from communicating any attributes. To focus on substantive consistency, we also include a temporal formula, a *purpose*, to compel communication, requiring, for example, that eventually a bank customer receives his account balance.

Def. A privacy policy θ is *consistent* with a purpose α if there exists a trace σ such that $\sigma \models \theta \wedge \alpha$.

Because the satisfiability of LTL formulas is a well-studied problem, we can apply a set of known algorithmic results [64, 30, 48] to evaluate consistency of privacy policies. By assuming our carrier sets are finite, we are able to rewrite universal and existential quantifiers as finite conjunctions and disjunctions in Propositional LTL (PLTL).

Theorem 1. *Policy consistency can be decided in PSPACE.*

Let β be an LTL formula expressing the knowledge evolution constraints on traces. The proof idea is to propositionalize $\theta \wedge \alpha \wedge \beta$ and decide its satisfiability in PSPACE (with respect to formula length and the size of the carrier sets). Although the worst-case complexity of satisfiability is PSPACE, there are efficient algorithms for several syntactic classes of formulas [30]. Furthermore, there are tools that work well in practice, such as the widely used SPIN model-checker [37].

4.1.2 Entailment

Another metric for evaluating a privacy policy is to compare it against another policy. For example, a hospital's privacy policy should not allow information flows prohibited by HIPAA.

Def. A privacy policy θ_1 *entails* a policy θ_2 if the LTL formula $\theta_1 \rightarrow \theta_2$ is valid over traces.

A hospital's privacy policy should entail HIPAA (which in turn should entail the norms of the societal health care context). Entailment generalizes the notion of *policy refinement* defined for EPAL in [13, 16]. These previous definitions are lattice-theoretic and require direct reasoning about upwards and downwards inheritance. Our simpler model-theoretic definition is made possible by representing policies as logical formulas that properly quantify over attributes. Here, policy entailment reduces to standard logical implication.

Theorem 2. *Policy entailment can be decided in PSPACE.*

This theorem is proved by observing that the formula $\theta_1 \rightarrow \theta_2$ is valid over traces just in case $\neg(\theta_1 \rightarrow \theta_2) \wedge \beta$ is not satisfiable, where β is an LTL formula for knowledge constraints. Deciding policy entailment for our policies is more difficult than for other privacy languages because we directly model temporal constraints instead of abstracting them into uninterpreted "obligations" (see Section 4.2.3).

Policy entailment also leads to notions of *policy combination*, as in [17, 12]. Entailment as implication gives rise to combination as logical conjunction and disjunction. This replaces the previous complex lattice-based definitions of other privacy languages. Policy combination is simpler in this framework because we represent policies by carefully constructed logical formulas and not by functions, as in XACML and EPAL. Representing policies as functions loses essential information about whether a requirement was inherited from another attribute. Representing policies as logical formulas retains the inheritance information, simplifying combination.

4.1.3 Compliance

Finally, we address the issue of compliance: given the sequence of past communications, does the policy permit a contemplated communication and, if so, what future requirements are incurred? This question has both a weak and a strong formulation. The weak formulation requires the contemplated action to satisfy all the necessary present conditions imposed by the policy. These necessary conditions are tracked using a standard PLTL construction called the tableau [48]. The tableau of a PLTL

formula is constructed by syntactically separating the present and future requirements. The future requirements characterize the sequences of actions that complete a finite trace to a satisfying infinite trace.

Def. Given a finite past history σ, an action a *weakly complies* with privacy policy θ if $\sigma \cdot a$ is a path in the tableau of θ that starts at an initial θ-atom. The *future requirements* of $\sigma \cdot a$ is the LTL formula ψ such that, for all traces σ',

$$\sigma' \models \psi \text{ if, and only if, } \sigma \cdot a \cdot \sigma' \models \theta.$$

Weak compliance ensures that each action taken by agents locally satisfies the privacy policy. However, a weakly compliant action could incur unsatisfiable future requirements. Weak compliance can be decided (and future requirements computed) using efficient techniques from LTL run-time verification [59].

Theorem 3. *Weak compliance and future requirements can be computed in polynomial time.*

In strong compliance, the information system ensures that agents can actually meet their future requirements while adhering to the policy. Note that previous privacy languages, such as EPAL, are able to determine only weak compliance because they lack a rich enough model of temporal conditions to determine the satisfiability of future requirements.

Def. Given a finite past history σ, an action a *strongly complies* with a privacy policy θ if there exists a trace σ' such that $\sigma \cdot a \cdot \sigma' \models \theta$.

Theorem 4. *Strong compliance can be decided in PSPACE.*

The complexity of checking strong compliance is in PSPACE because it involves checking for satisfiability. However, because the typical use of this algorithm will be at each point in a trace (for example in a hospital information system), it is natural to ask whether it is possible to reduce the complexity of checking whether each action is compliant by doing more work at the beginning of the execution. If weak compliance for a policy implies strong compliance, an information system need only require weak compliance (which can be computed efficiently) in order to achieve strong compliance.

Model	Sender	Recipient	Subject	Attributes	Past	Future	Combination
RBAC	Role	Identity	×	×	×	×	•
XACML	Flexible	Flexible	Flexible	○	×	○	•
EPAL	Fixed	Role	Fixed	•	×	○	×
P3P	Fixed	Role	Fixed	•	○	×	○
LPU	Role	Role	Role	•	•	•	•

Figure 4.1: Comparison of various privacy languages. The symbol × indicates the feature is absent from the language, ○ indicates partial or limited functionality, and • indicates the feature is fully functional. Note, [12] gives an extension of EPAL that is closed under combination.

Theorem 5. *Given a privacy policy θ, it can be decided whether weak compliance for θ implies strong compliance in exponential space.*

The main idea behind the proof is to construct the automaton for θ and check that there is a path from every reachable state to a strongly connected component.

4.2 Comparison with Other Models

In this section, we compare LPU with traditional Role-Based Access Control (RBAC), the eXtensible Access Control Markup Language (XACML), the Enterprise Privacy Authorization Language (EPAL), and the Platform for Privacy Preferences (P3P). LPU generalizes these existing models in two key ways. First, LPU includes an extensive language for defining temporal conditions, improving the rudimentary future "obligations" of XACML and EPAL. Second, LPU correctly handles temporal conditions associated with negative norms (denying rules). Temporal conditions can be attached to denying rules in XACML and EPAL, but the resulting semantics are murky. Our findings are summarized in Figure 4.1.

4.2.1 Role-Based Access Control

RBAC (see, for example, [20]) is an access control model in which access rights are specified in terms of roles. LPU generalizes RBAC by specifying more parameters by role, containing a notion of attribute and data subject, and including temporal

conditions. RBAC can express policies about arbitrary actions, whereas LPU, as currently formulated, is concerned solely with communication actions. LPU replaces the "object" of RBAC with a recipient principal, enabling the "actee" (object or recipient) to be specified by a role. RBAC rules are positive and negative norms of the following forms, respectively:

$$\begin{aligned} \text{Allow:} \quad & \mathsf{inrole}(p_1, \hat{r}_1) \wedge (p_2 = \hat{p}_2) \\ \text{Deny:} \quad & \mathsf{inrole}(p_1, \hat{r}_1) \wedge (p_2 = \hat{p}_2) \rightarrow \bot \end{aligned}$$

Notice RBAC lacks the subject q and attribute t. Temporal conditions are also absent. "Deny" rules are expressible in LPU by negative norms with $\bot$, the unsatisfiable formula.

The key reason RBAC is insufficient for privacy is that it lacks the notion of an attribute. Suppose a doctor reads a patient's medical file and then sends an email to his broker. From an RBAC perspective, nothing untoward has occurred. Both actions, reading the file and sending the email, are (presumably) permitted by the policy. However, a privacy breach has occurred if the doctor includes sensitive medical information about another patient in his email. To distinguish the appropriate from the inappropriate, it is essential to recognize the attributes communicated by each action. In other words, RBAC is insufficient for privacy because it lacks the "$\mathsf{contains}$" relation.

Several access control languages, such as Binder [31] and RT [46], extend RBAC using Datalog. Typically, these languages use only positive rules and contain neither temporal conditions nor a notion of the subject of a piece of information. Cassandra [18], a sophisticated access control language with denying rules, has been applied to electronic health records in the United Kingdom. In that study, consent was captured through role activation: a patient consents to treatment by activating a "consent-to-treatment" role. Future temporal constraints, as well as notions of computing attributes, are absent.

4.2.2 Extensible Access Control Markup Language

The Extensible Access Control Markup Language [7] is a flexible language for expressing access control policies. XACML's extension mechanism enables XACML to capture a wide variety of access control constructs. To make meaningful statements about the expressiveness of XACML, we restrict our attention to policies expressible by simple extensions to the base XACML language. In particular, we abstract XACML's targets as elements of a Boolean algebra over a set of requests and consider only the built-in combination algorithms.

XACML lacks first-class temporal conditions. When an XACML policy reaches a policy judgment, it can include in its response an "obligation," a symbol to be interpreted at the point of policy enforcement. These uninterpreted symbols can be used to represent future requirements. Obligations, however, prevent the semantics of an XACML policy from being fully specified by the policy itself (because the policy relies on the surrounding environment to give the obligations meaning). Past conditions can also be expressed in XACML by encoding state information into the "request context," additional information passed to the policy evaluation engine. However, using this feature to capture more complex states than "opt-in" and "opt-out" is awkward.

XACML is unable to correctly capture attributes [6], especially in connection with denying rules (negative norms). The difficulty arises because XACML conceives of a policy as a *function* from requests to responses. XACML policies are structured as combinations of simple subpolicies, where combination is computed point-wise on the functions represented by the subpolicies. This fails for attributes because the effect of combination can be non-local (due to "upward" inheritance). The combined response for two policies on a request is not necessarily determined by the responses of the subpolicies on *that* request. LPU avoids this by representing and combining policies logically.

4.2.3 The Enterprise Privacy Authorization Language

The Enterprise Privacy Authorization Language is expressly designed for expressing enterprise privacy policies [11, 63]. EPAL policies are concerned with a single sender (the enterprise itself) and a single subject role [41]. EPAL has the same limitations as XACML on its temporal conditions.

EPAL requests are elements of a Cartesian product of trees representing roles, attributes, purposes, and actions. The "role" coordinate represents the role of the recipient. The "purpose" coordinate is not captured directly in LPU. However, these purposes can be simulated in LPU (see below). Finally, EPAL policies are concerned with general actions, not just with communication actions, as in RBAC. With the exception of purposes and non-communication actions, LPU captures EPAL policies using positive and negative norms of the following forms, respectively:

$$(p_1 = \hat{p}_1) \wedge \mathsf{inrole}(p_2, \hat{r}_2) \wedge (t \in \hat{t}) \wedge \hat{o}$$
$$(p_1 = \hat{p}_1) \wedge \mathsf{inrole}(p_2, \hat{r}_2) \wedge (t \in \hat{t}) \rightarrow \bot$$

The sender agent $\hat{p}_1$ is fixed for every norm in a single policy. The symbol $\hat{o}$ is a propositional letter that represents an uninterpreted future "obligation," similar to how obligations are represented in XACML. EPAL structures these obligations with a subsumption relation.

LPU improves on EPAL obligations in two respects. First, obligations are expressed in temporal logic (as in [39]), the same logic as the policy itself. Thus, tools can interpret temporal conditions and determine, for example, whether or not it is possible for an agent to discharge his or her future obligations while adhering to the policy. Second, the temporal conditions in LPU can speak about the past as well as the future, enabling policies that permit information flows in virtue of past actions. In LPU, the subsumption relation on temporal conditions arises naturally as the logical implication of temporal formulas. Future obligations in the form of a list of future actions that must be performed are present in the policy specification language Ponder [29]. These obligations are richer than EPAL's uninterpreted obligations, but are restricted to **F** conditions, failing to capture the **FG** condition in COPPA norm.

EPAL policy authors can attach obligations to denying rules. However, the semantics of such obligations are dubious: the policy engine responds that a contemplated action both is denied and incurs an obligation, but is the obligation incurred if the requesting agent does not perform the contemplated action? LPU resolves this difficulty by weakening the notion of a denying rule to that of a negative norm, a formula of the form $\varphi \rightarrow \psi$. Negative norms do not forbid actions described by φ, but instead forbid actions described by φ that violate the temporal condition ψ. Complete prohibitions can be expressed by instantiating ψ with $\bot$.

EPAL purposes in LPU. In EPAL, each action is conducted for some purpose. An EPAL policy can permit an action for a particular purpose and also deny the same action for a different purpose. For example, a health web site might be permitted to analyze visitor health information for medical purposes, but might not be permitted to analyze the same health information for marketing purposes. LPU can capture this notion by decomposing large agents into several smaller agents, one for each purpose. For example, the monolithic health web site could be decomposed into a medical agent and a marketing agent. EPAL purposes could then be expressed in LPU by restricting communication between the constituent agents.

4.2.4 The Platform for Privacy Preferences

The Platform for Privacy Preferences (P3P) is a privacy language intended for use by web site operators in informing their visitors of their data practices [27, 58]. P3P contains only positive norms and restricted temporal conditions. A single P3P policy is restricted to a single sender (the web site) and a single subject role (a web site visitor). These restrictions impair the use of P3P as a general-purpose privacy language. For example, to state that a web site complies with COPAA, a web site operator must employ a P3P extension [26] and make the policy statement `COPPA status="compliant"`. Temporal conditions in P3P are limited to `opt-in`, `opt-out`, and `true`. P3P statements correspond to positive norms of the following form:

$$(p_1 = \hat{p}_1) \wedge \mathsf{inrole}(p_2, \hat{r}_2) \wedge \mathsf{inrole}(q, \mathit{visitor}) \wedge (t \in \hat{t}) \wedge \psi$$

where ψ represents "opt-in," "opt-out," or no temporal condition. The lack of negative norms simplifies P3P at the cost of expressiveness. The fixed form of the opt-in and opt-out conditions is restrictive, preventing even minor variations such as the parental "grant-consent" and "revoke-consent" idiom found in COPPA.

P3P provides for privacy preference languages that a web surfer can use to filter out web sites with unwanted data practices. These preference languages highlight another difference between P3P and LPU: all P3P policies inhabit a single global context. A web surfer cannot specify different preferences for medical web sites than for financial web sites. This forces web surfers to resort to a "lowest common denominator" preference. Both the preference languages APPEL [27] and XPref [3] can express negative preferences, but such preferences are not respected in the full P3P system [16].

4.3 Expressing Privacy Regulations

In this section, we evaluate the expressiveness of the logic of privacy and utility by showing how to represent some commonly discussed privacy regulation. We intend our framework to express organizational privacy policies as well as regulation, but we focus on regulation in this chapter for concreteness. We can capture most of the privacy notions embedded in the laws we examine, and conversely the laws we examine exercise most of the features of our model. We regard this as evidence that the logic has roughly the correct level of expressiveness to represent generally accepted notions of privacy.

We consider three pieces of regulation: the Health Insurance Portability and Accountability Act (HIPAA), the Children's Online Privacy Protection Act (COPPA), and the Gramm–Leach–Bliley Act (GLBA). The distinction between positive and negative norms surfaces in the different approaches taken by these laws. At a high level, HIPAA forbids disclosure of protected health information except in enumerated capacities, whereas COPPA and GLBA forbid enumerated information flows. Temporal conditions attached to negative norms are common in COPPA and GLBA. The mishandling of negative temporal conditions in other frameworks hampers their

ability to capture these privacy laws correctly, whereas LPU is able to capture both flavors of policy in a unified logical framework.

4.3.1 The HIPAA Privacy Rule

The HIPAA Privacy Rule regulates the transmission of "protected health information" (*phi*), by *covered entities*, such as hospitals, doctors, and insurance companies [54]. HIPAA largely forbids the disclosure of health information except to individuals or organizations acting in particular roles. HIPAA contains many privacy provisions, most of which can be expressed directly as positive transmission norms. We present a few representative examples.

$$\mathsf{inrole}(p_1, \textit{covered-entity}) \wedge \mathsf{inrole}(p_2, \textit{individual}) \wedge (q = p_2) \wedge (t \preceq \textit{phi}) \tag{4.1}$$

The norm above allows a covered entity to communicate *phi* about an individual to that individual. This norm allows Dr. Alice to show Bob an x-ray of his broken leg. It does not allow, however, Dr. Alice to show Bob's x-ray to Carol. Moreover, it does not permit x-ray technician Debbie to give the x-ray to Dr. Alice. For that communication, HIPAA provides another norm:

$$\mathsf{inrole}(p_1, \textit{covered-entity}) \wedge \mathsf{inrole}(p_2, \textit{provider}) \wedge \mathsf{inrole}(q, \textit{patient}) \wedge (t \preceq \textit{phi})$$

Dr. Alice is not only a covered entity, but also, more specifically, a health care *provider*, someone directly involved in the care of a patient. Here, Debbie plays the role of covered entity and is permitted to give Bob's x-ray to Dr. Alice (Bob plays the role of patient).

Although the bulk of HIPAA consists of positive norms dealing with the attribute *phi*, HIPAA does contain a negative norm dealing with a component of *phi*: psychotherapy notes. The rule provides special protection for the disclosure of psychotherapy notes, even to the individual whom the notes are about. In particular,

HIPAA contains a negative norm that prevents a covered entity from disclosing psychotherapy notes to the subject of the notes without the prior approval of a psychiatrist:

$$\begin{aligned}&\mathsf{inrole}(p_1, \textit{covered-entity}) \wedge \mathsf{inrole}(p_2, \textit{individual}) \wedge (q = p_2) \wedge \\ &\qquad (t \preceq \textit{psychotherapy-notes}) \rightarrow \mathbf{F}^{-} \exists p : P.\, \mathsf{inrole}(p, \textit{psychiatrist}) \wedge \\ &\qquad\qquad \mathsf{send}(p, p_1, \textit{approve-disclose-psychotherapy-notes}) \quad (4.2)\end{aligned}$$

The interplay between the positive and negative norms is subtle. One positive norm (4.1) permits the disclosure of psychotherapy notes, but a negative norm (4.2) prevents it (unless approval is obtained). These norms are not contradictory because the positive norm does not require the disclosure. Moreover, even after approval is received (satisfying the negative temporal condition), the covered entity would not be allowed to disclose the notes without the positive norm.

HIPAA contains specific norms for directories of facilities such as hospitals. Specifically, it provides that a covered entity may "disclose the individual's [general] condition and location within the facility to anyone asking for the individual by name" [54]. This can be expressed as a positive norm:

$$\begin{aligned}&\mathsf{inrole}(p_1, \textit{covered-entity}) \wedge \mathsf{inrole}(p_2, \textit{individual}) \wedge \\ &\qquad \mathsf{inrole}(q, \textit{individual}) \wedge (t \preceq \textit{condition-and-location}) \wedge \\ &\qquad\qquad \mathbf{F}^{-} \exists m' : M.\, \mathsf{send}(p_2, p_1, m') \wedge \mathsf{contains}(m', q, \textit{name})\end{aligned}$$

The rule also contains a provision allowing members of the clergy to obtain directory information. This is expressed in the norm below, where *directory-information* is an attribute that contains (formally can be used to compute) the individual's name, general condition, religious affiliation, and location within the facility.

$$\begin{aligned}&\mathsf{inrole}(p_1, \textit{covered-entity}) \wedge \mathsf{inrole}(p_2, \textit{clergy}) \wedge \\ &\qquad\qquad \mathsf{inrole}(q, \textit{individual}) \wedge (t \preceq \textit{directory-information})\end{aligned}$$

The use of such information by the clergy is subject to further norms, but this is outside the scope of HIPAA.

De-identified Health Information. Most of the privacy provisions of the HIPAA Privacy Rule can be expressed using norms of transmission. Some provisions, however, are outside the LPU model. In particular, HIPAA provides that covered entities can disclose "de-identified health information" without regard to the other provisions of the rule. In our formalization of contextual integrity, every attribute is "identified" in virtue of being associated with an agent. Although we have not examined this in detail, we expect that an extended model with group attributes (attributes about a set of agents) could capture de-identified attributes. The relation between individual attributes and de-identified attributes has been studied extensively (e.g., [4, 33, 66]).

4.3.2 Children's Online Privacy Protection Act (COPPA)

COPPA protects the personal information children communicate to web sites [35]. It differs from HIPAA in two ways. First, COPPA does not contain an enumeration of positive norms. Instead, it contains two negative norms that restrict otherwise permissible flows of information. Second, temporal conditions play a central role in COPPA. The temporal conditions require web sites that collect protected information from children to respond in a particular way to messages from parents.

COPPA applies when a *child* sends individually identifiable information, *protected-info*, about him- or herself to a *web site* operator over the Internet. The two central negative norms of COPPA have a similar form, differing only in their temporal conditions. Whenever a child sends a web site his or her protected information, the web site operator is bound to follow both temporal conditions, one requiring "parental consent" and another providing a "right of access."

$$\begin{aligned}\mathsf{inrole}(p_1, \mathit{child}) \wedge \mathsf{inrole}(p_2, \mathit{web\text{-}site}) \wedge (q = p_1) \wedge (t \preceq \mathit{protected\text{-}info}) \rightarrow \\ \exists p : P.\, \mathsf{inrole}(p, \mathit{parent}) \wedge \neg\, \mathsf{send}(p, p_2, \mathit{revoke\text{-}consent})\mathbf{S} \\ (\mathsf{send}(p, p_2, \mathit{grant\text{-}consent}) \wedge \mathbf{F}^{-}\, \mathsf{send}(p_2, p, \mathit{privacy\text{-}notice}))\end{aligned}$$

The negative norm above requires web site operators to obtain parental consent before collecting protected information from children. When a child sends protected information to a web site, a *parent* must have previously received a privacy notice from the web site operator, granted consent to the web site operator, and not since revoked that consent. Notice the strong form of "since" is required here to ensure that the parent actually granted consent.

$$\begin{aligned}\mathsf{inrole}(p_1, \mathit{child}) \wedge \mathsf{inrole}(p_2, \mathit{web\text{-}site}) \wedge (q = p_1) \wedge (t \preceq \mathit{protected\text{-}info}) \rightarrow \\ \mathbf{G}\forall p : P.\, \mathsf{inrole}(p, \mathit{parent}) \wedge \mathsf{send}(p, p_2, \mathit{request\text{-}information}) \rightarrow \\ \mathbf{F}(\mathsf{send}(p_2, p, \mathit{privacy\text{-}notice}) \wedge \mathsf{send}(p_2, p, m))\end{aligned}$$

The negative norm above contains a temporal condition that requires web site operators to furnish parents with a privacy notice describing their information practices as well as the specific information they have collected from the child. This reactive condition is easily expressed using the **GF** modality.

The first temporal condition is concerned with the past, that a parent has given consent, whereas the second condition is concerned with the future, that the web site operator reacts correctly to parental requests. COPPA requires web site operators to verify that they are indeed communicating with one of the child's parents before disclosing the child's protected information. Such verification is represented in our model by assigning the role *parent* to the appropriate agents. COPPA also requires the operator to delete protected information in its possession upon receiving *revoke-consent*. Our model does not capture "forgetting" actions, but such actions can be included in the model, at the cost of complexity.

4.3.3 Gramm–Leach–Bliley Act (GLBA)

The Financial Modernization Act of 1999, commonly referred to as the Gramm–Leach–Bliley Act or GLBA, contains privacy provisions limiting how financial institutions can handle the non-public personal information, *npi*, of their customers and

consumers [36]. Broadly, GLBA requires financial institutions to inform their customers of their privacy practices and to allow customers to "opt-out" of some kinds of information disclosures.

Financial *institutions* are required to send their *customers* privacy notices every year as long the customer relationship lasts. Without numerical notions of time, LPU cannot express that the notices must be delivered annually. Instead, the negative norm below requires institutions to periodically send privacy notices.

$$\mathsf{inrole}(p_1, \mathit{customer}) \wedge \mathsf{inrole}(p_2, \mathit{institution}) \wedge (q = p_1) \wedge (t \preceq \mathit{npi}) \rightarrow \\ \mathbf{F}\,\mathsf{send}(p_2, p_1, \mathit{privacy\text{-}notice})\mathbf{W}\neg\,\mathsf{inrole}(p_1, \mathit{customer})$$

In addition to a customer role, GLBA distinguishes a *consumer* role. GLBA's requirements on interacting with consumers are less strict than its requirements on interacting with customers. Institutions are required to notify consumers of their privacy practices only if they share the consumer's *npi* with *non-affiliated* companies, and they may do so before or after the disclosing *npi*. The negative norm below makes essential use of the three different roles (sender, recipient, and subject), as well as both past and future modalities in its temporal condition.

$$\mathsf{inrole}(p_1, \mathit{institution}) \wedge \mathsf{inrole}(p_2, \mathit{non\text{-}affiliate}) \wedge \mathsf{inrole}(q, \mathit{consumer}) \wedge (t \preceq \mathit{npi}) \rightarrow \\ \mathbf{F}\,\mathsf{send}(p_1, q, \mathit{privacy\text{-}notice}) \vee \mathbf{F}^{-}\,\mathsf{send}(p_1, q, \mathit{privacy\text{-}notice})$$

Both consumers and customers can "opt-out" of the sharing of *npi* with non-affiliated companies. The norm below expresses the provision for consumers, and GLBA also contains an analogous non-affiliate opt-out norm for customers.

$$\mathsf{inrole}(p_1, \mathit{institution}) \wedge \mathsf{inrole}(p_2, \mathit{non\text{-}affiliate}) \wedge \mathsf{inrole}(q, \mathit{consumer}) \wedge (t \preceq \mathit{npi}) \rightarrow \\ \neg\mathbf{F}^{-}\,\mathsf{send}(q, p_1, \mathit{opt\text{-}out\text{-}of\text{-}non\text{-}affiliate})$$

Consumers and customers also have the option of opting out of some kinds information sharing between institutions and their *affiliates*, such the sharing of credit reports and

application information. The norm below expresses the provision, and GLBA contains a similar norm for application information. GLBA contains some exceptions to these norms, but we omit those here for clarity.

$$\mathsf{inrole}(p_1, \textit{institution}) \wedge \mathsf{inrole}(p_2, \textit{affiliate}) \wedge \mathsf{inrole}(q, \textit{consumer}) \wedge (t \preceq \textit{credit-report}) \rightarrow \neg \mathbf{F}^{-}\, \mathsf{send}(q, p_1, \textit{opt-out-of-affiliate})$$

Much of the consternation about GLBA revolves around the complex definition of which companies are affiliates and what precisely constitutes non-public personal information [34]. Our formalization of these norms sidesteps these issues by taking the role *affiliate* and the attribute *npi* to be defined exogenously: the judgments as to which companies are *affiliates* and which communications contain *npi* are made in the preparation of a trace history. The machinery of the model then classifies this trace history as respecting or as not respecting the norms of transmission.

The use of negative norms in the expression of GLBA is essential: replacing the negative norms with their positive duals fails to express GLBA. Consider Alice, who is both a customer and a consumer of financial institution FirstCyber. In the negative formulation of GLBA, if she sends *npi* to FirstCyber, FirstCyber must periodically send her privacy notices. In the attempted positive formulation, however, if she sends *npi* to FirstCyber, FirstCyber need not periodically send her privacy notices. The disjunctive character of positive norms enables FirstCyber to choose, for each communication, whether to regard Alice as a customer or as a consumer. In the negative formulation, the conjunctive character of the negative norms requires FirstCyber to treat Alice as both a customer and a consumer.

Chapter 5

Utility and Business Processes

In this chapter, we expand the discussion of the Logic of Privacy and Utility (LPU) to encompass the utility goals of an organization. Our notion of utility makes use of the strategy quantifier in the logic to determine if a coalition of agents can bring about some useful outcome from a privacy-constrained workflow. We consider both the design-time evaluation of an organizational workflow and the run-time auditing of the workflow's execution. Portions of this chapter appear in [15].

5.1 Workflows and Responsibility

A workflow is a division of responsibility among human agents and a mechanical workflow engine. By assigning responsibilities to human agents, workflows can achieve privacy and utility goals beyond the capabilities of mechanical systems. We develop general design principles and desiderata using abstract workflows, which are collections of formulas specifying agent and engine responsibilities, and specific auditing algorithms for practical workflows like MyHealth based on communication graphs.

Abstract Workflows. Abstractly, a *workflow* is an LTL sentence φ together with an LTL formula $\varphi_r(x)$ for each role r. The mechanical workflow engine is responsible for achieving φ and a human agent p in role r is responsible for achieving $\varphi_r(p)$. Because the workflow engine can only prevent communication initiated by human

agents, it can enforce only safety properties (properties that fail at a finite time). Also, as a mechanical agent, the workflow engine must authorize communication based on message tags (and not the inaccessible message contents).

Def. A workflow is *feasible* if φ is a safety formula[1] without the $\mathsf{contains}$ predicate and $\mathcal{G} \models \forall p.\varphi \wedge \mathsf{inrole}(p, r) \rightarrow \langle\langle p \rangle\rangle \varphi_r(p)$, for every role r.

In a feasible workflow, agents have local strategies for living up to their responsibilities and, thus, know which actions or inaction are responsible based on their observations of previous actions (assuming the workflow engine is functioning property). Moreover, if an agent is responsible for sending a message, feasibility ensures that the agent will be able to send the message.

Graph-based Workflows. The MyHealth workflow depicted in Section 6.1 is a practical kind of workflow based on a *workflow graph*, a set of nodes $\mathcal{R}$, the roles of the workflow, and a function T such that $T(r_1, r_2) \subseteq \mathcal{T}$ for all $r_1, r_2 \in \mathcal{R}$, where $\mathcal{T}$ is the set of attributes. The workflow engine permits a message m to be sent from an agent in role r_1 to an agent in role r_2 if, and only if, $\text{tags}(m) \subseteq T(r_1, r_2)$. The responsibility of the workflow engine is the conjunction over $r_1, r_2 \in \mathcal{R}$ of

$$\mathbf{G}\forall p_1, p_2, q, m.\, \mathsf{inrole}(p_1, r_1) \wedge \mathsf{inrole}(p_2, r_2) \wedge \mathsf{send}(p_1, p_2, m) \wedge \mathsf{tagged}(m, q, t) \rightarrow t \in T(r_1, r_2).$$

The workflow engine is stateless and distributed because the decision about whether to block a communication is based only on the current action, not on past or non-local actions. The responsibilities of human agents are divided into two parts: *tagging responsibilities* and *progress responsibilities*. If an agent is permitted to send a particular type of information, the agent is responsible for attaching tags for that type to all messages the agent sends that contain that type of information. The tagging

[1] There is a standard syntactic definition of safety formulas [48].

responsibility for role r is the conjunction over $t \in T(r, *)$ of

$$\mathbf{G}\forall p_2, q, m.\, \mathsf{send}(x, p_2, m) \wedge \mathsf{contains}(m, q, t) \rightarrow \mathsf{tagged}(m, q, t).$$

In MyHealth, nurses are responsible for attaching *health-question*, *appt-request*, and *health-answer* tags to messages. Tagging constraints are feasible because agents are free to select the tags for messages they send. Progress responsibilities require agents to eventually send messages. Progress responsibilities are essential in achieving the liveness requirements of privacy goals (such as notification requirements) and in achieving utility goals. A progress responsibility is a formula of the form $\mathbf{G}\forall \vec{x}.\psi \rightarrow \mathbf{F}\theta$, where ψ and θ are past formulas. For example, in MyHealth doctors have the following progress responsibility.

$$\mathbf{G}\forall p, q, m.\, \mathsf{send}(p, x, m) \wedge \mathsf{contains}(m, q, \textit{health-question}) \rightarrow \\ \mathbf{F}\exists m'.\, \mathsf{contains}(m, q, \textit{health-answer}) \wedge \mathsf{send}(x, q, m')$$

We do not make explicit how m' depends on m because m' is created by a human agent, not by a piece of code. A more detailed model might bind the health answer to the question by way of a transaction identifier or a more explicit notion of causality.

5.2 Privacy and Utility in Workflow Design

Organizations design workflows to accomplish particular tasks while complying with regulation and privacy policies. This section contains algorithms for determining if a workflow design achieves these utility and privacy goals provided all agents act responsibly. If several workflows achieve privacy and utility, many privacy advocates [49] recommend deploying a workflow that minimizes the information disclosed to agents. We formulate a design criterion called *minimality* that makes this intuition precise.

5.2.1 Privacy

Many of the privacy provisions found in US regulation can be expressed in LPU (see Section 4.3). For example, one of MyHealth's privacy goals, derived from HIPAA, is that no patient should learn answers to another patient's health questions.

$$\mathbf{G}\forall p_1, p_2, q, m.\, \mathsf{send}(p_1, p_2, m) \wedge \mathsf{inrole}(p_2, \mathit{patient}) \wedge \mathsf{contains}(m, q, \mathit{health\text{-}answer}) \rightarrow q = p_2$$

Responsible executions of a workflow $(\varphi, \varphi_{\mathcal{R}})$ is characterized in LTL as follows.

$$\mathit{agents\text{-}responsible} \equiv \varphi \wedge \bigwedge_{r \in \mathcal{R}} \forall p.\, \mathsf{inrole}(p, r) \rightarrow \varphi_r(p)$$

If $\mathcal{G} \models_{\text{LTL}} \mathit{agents\text{-}responsible} \rightarrow \mathit{privacy\text{-}policy}$, then responsible agents will achieve a privacy goal *privacy-policy*. However, this requirement imposes a design constraint only on agent responsibilities, not on the responsibility of the workflow engine. When the tags are correct, the workflow engine is capable of enforcing safety properties but is incapable of enforcing liveness properties. The executions of the workflow with correct tags is expressed by the LTL formula

$$\mathit{tags\text{-}correct} \equiv \varphi \wedge \forall p_1, p_2, q, m, t.\, \mathsf{send}(p_1, p_2, m) \rightarrow (\mathsf{tagged}(m, q, t) \leftrightarrow \mathsf{contains}(m, q, t)).$$

A workflow achieves a privacy goal if responsible agents fulfill the privacy goal and the workflow engine fulfills the safety component of the privacy policy when operating with correct tags.

Def. A workflow *achieves* a privacy goal *privacy-policy* if

$$\mathcal{G} \models_{\text{LTL}} \mathit{tags\text{-}correct}\, \mathbf{U}\, \mathit{agents\text{-}responsible} \rightarrow \mathit{privacy\text{-}policy}.$$

Algorithmically, we consider the propositional case in which there are finite numbers of agents, messages, attributes, and roles.

Theorem 6. *It can be decided whether a workflow achieves a privacy goal using space polynomial in the description of the workflow and the number of agents.*

In practice, if an organization has a large number of agents, we suggest the standard practice of evaluating whether a workflow achieves privacy on a smaller model in which the model-checking problem is tractable.

5.2.2 Utility

A workflow is useful if some execution accomplishes a task, although that task need not be accomplished in every execution. A workflow achieves a utility goal if some of the agents have a strategy for accomplishing the task. Formally, a utility goal for agents in a vector of roles $\vec{r}$ is a sentence of the form

$$\forall \vec{p}.\, \mathsf{inrole}(\vec{p}, \vec{r}) \rightarrow \langle\langle \vec{p} \rangle\rangle \psi,$$

where ψ, the task, is an LTL formula. For example, one utility requirement in MyHealth is that patients can receive answers to their health questions.

$$\forall p.\, \mathsf{inrole}(p, \mathit{patient}) \rightarrow \langle\langle p \rangle\rangle \mathbf{F} \exists p_1, m.\, \mathsf{send}(p_1, p, m) \wedge \mathsf{contains}(m, p, \mathit{health\text{-}answer}).$$

A workflow achieves a utility goal *utility-goal* if the agents have a strategy for responsibly accomplishing their task if the other agents act responsibly, formalized as the ATL* entailment $\mathcal{G} \models \mathit{agents\text{-}responsible} \rightarrow \mathit{utility\text{-}goal}$.

In general, deciding whether a workflow satisfies this condition is undecidable [5] because $\mathcal{G}$ is a game of imperfect information: an agent's strategy is based only on the messages that the agent has sent or received, not on messages exchanged between other agents. For example, a patient's strategy for obtaining an answer to his or her health question cannot depend on whether messages have been exchanged between a doctor and a nurse. To avoid undecidability, we use a sound approximation based on local communication games. This approximation has proven adequate for the examples we have examined thus far.

Local Communication Game. Instead of checking the formula in the full game $\mathcal{G}$, we check the formula in the *local communication game* for agent p, $\mathcal{G}_p$, a game of perfect information where we can apply the standard model-checking algorithm for ATL*. Checking the formula in the local model is sound (but not complete) for a particular syntactic class of formulas that includes utility goals. The local communication game $\mathcal{G}_p$ is defined from the full communication game $\mathcal{G}$ by way of $\sim_p$, the smallest equivalence relation such that $\kappa_1 \sim_p \kappa_2$ if $\kappa_1 \xrightarrow{a} \kappa_2$, where κ_1 and κ_2 are knowledge states of $\mathcal{G}$ and a is invisible to p.

Proposition 7. *$\hat{\kappa} \sim_p \kappa$ implies $\hat{\kappa}_p = \kappa_p$, for all knowledge states $\hat{\kappa}$ and κ.*

The states of $\mathcal{G}_p$ are the equivalence classes $[\kappa]_p$ of knowledge states κ of $\mathcal{G}$ under $\sim_p$. An action is available in $[\kappa]_p$ if p considers the action possible. For all sets of actions A, $[\kappa]_p \xrightarrow{A} [\kappa']_p$ if there exist $\hat{\kappa} \sim_p \kappa$ and $\hat{\kappa}' \sim_p \kappa'$ with $\hat{\kappa} \xrightarrow{A} \hat{\kappa}'$.

Lemma 8. *For all sets of actions A and all knowledge states κ, κ_1, and κ_2,*

$$[\kappa]_p \xrightarrow{A} [\kappa_1]_p \text{ and } [\kappa]_p \xrightarrow{A} [\kappa_2]_p \text{ implies } \kappa_1 \sim_p \kappa_2.$$

Proof. It suffices to prove the statement for A containing a single action. If p is not the recipient of a send message, then $[\kappa_1]_p = [\kappa_2]_p = [\kappa]_p$. If p is a recipient, then $\kappa_1(p)$ and $\kappa_2(p)$ and the other agents can invisibly exchange messages to equalize their knowledge as well, yielding $\kappa_1 \sim_p \kappa_2$. □

The main idea of the proof is contained in Lemma 9, which connects the truth value of visible formulas in $\mathcal{G}_p$ with their truth value in $\mathcal{G}$. The atomic formulas *visible* to agent p are $\mathsf{send}(p, *, *)$, $\mathsf{send}(*, p, *)$, $\mathsf{inrole}(*, *)$, $\mathsf{contains}(*, *, *)$, and $\mathsf{tagged}(*, *, *)$, where $*$ is an arbitrary term. A formula is *visible* to p if, and only if, all its atomic formulas are visible to p. The truth values of formulas visible to p are determined by the view of p.

Lemma 9 (Soundness)**.** *For all LTL formulas φ that are visible to p,*

$$\mathcal{G}_p \models \langle\langle p \rangle\rangle \varphi \text{ implies } \mathcal{G} \models \langle\langle p \rangle\rangle \varphi.$$

Proof Sketch. If p has a strategy to force φ in $\mathcal{G}_p$, then p must also have a strategy to force φ in $\mathcal{G}$ because the set of moves available to p's opponents in $\mathcal{G}$ is a subset of the set of moves available to p's opponents in $\mathcal{G}_p$. Fix a strategy Γ_p for p that forces φ in $\mathcal{G}_p$. For every finite p-view $\pi \upharpoonright p$ of $\mathcal{G}$, let $\Gamma(\pi \upharpoonright p) = \Gamma_p([\pi]_p)$. Γ is a local strategy for p in $\mathcal{G}$ that forces φ because φ is visible to p. □

We can now state a sound algorithm for determining whether $\mathcal{G} \models \psi \rightarrow \langle\langle p \rangle\rangle \varphi$ for propositional LTL formulas ψ and φ.

Algorithm 10. *Let G be the labeled transition system of $\mathcal{G}$ and let A^ψ be the Büchi automata for ψ (see, for example, [48]). Construct G^ψ as the conjunction of the automata G and A^ψ and use it as the frame of the concurrent game structure $\mathcal{G}^\psi$. Let $\mathcal{G}^\psi_p$ be the local communication game. Report true if $\mathcal{G}^\psi_p \models \langle\langle p \rangle\rangle \varphi$ (determined using standard ATL* model checking, i.e. [5]).*

Theorem 11. *Algorithm 10 is sound for deciding whether*

$$\mathcal{G} \models \textit{agents-responsible} \rightarrow \textit{utility-goal}.$$

Proof Idea. Intuitively, Algorithm 10 uses deduction to translate the problem into $\mathcal{G}, \textit{agents-responsible} \models \textit{utility-goal}$ and then applies Lemma 9 to use the standard ATL* model-checking algorithm. Each step in the algorithm preserves soundness, so the entire algorithm is sound. □

5.2.3 Minimal Workflow

The Markle Foundation [49], among many others, advocates the principle of *minimum necessary disclosure* for systems processing personal information. Under this principle, each agent should receive only the information needed for the workflow to achieve its utility goals. We begin to make this notion precise by inducing a partial order relation on workflows.

Def. One workflow $W_1 = (\varphi_1, \varphi_\mathcal{R})$ is *at least as restrictive as* another workflow $W_2 = (\varphi_2, \varphi_\mathcal{R})$, written $W_1 \leq W_2$, if $\mathcal{G} \models \varphi_1 \rightarrow \varphi_2$.

If $W_1 \leq W_2$, the workflow engine permits agents to learn no more information in W_1 than in W_2. For graph-based workflows, $W_1(\mathcal{R}, T_1)$ is at least as restrictive as $W_2(\mathcal{R}, T_2)$ if $T_1(r_1, r_2) \subseteq T_2(r_1, r_2)$ for all $r_1, r_2 \in \mathcal{R}$. In this initial formulation, workflows are only comparable under $\leq$ if they have the same agent responsibilities.

Proposition 12 (Monotonicity)**.** *For all workflows W_1 and W_2, if $W_1 \leq W_2$ and W_2 achieves a privacy goal, then W_1 also achieves the privacy goal.*

The ordering $\leq$ is conservative in the sense that if two workflows are related by $\leq$, then the smaller one discloses less information, but if two workflows are incomparable under $\leq$, one might still disclose less information. There does not appear to be a direct connection between this ordering and utility goals because whether an agent has a strategy to achieve a goal might be helped or hindered by strengthening the engine responsibility.

Def. A workflow W is *minimal* for a utility goal if W achieves the utility goal and all feasible workflows $W' < W$ fail to achieve the utility goal (where $W' < W$ if $W' \leq W$ and $W \not\leq W'$).

Minimal workflows provide the strongest privacy for a given utility goal, as advocated by the principle of minimum necessary disclosure. For abstract worflows, minimal workflows (as defined) fail to exist because the engine responsibility φ of a candidate minimal workflow can always be strengthened by conjoining extraneous conditions. This definition is useful because it provides a precise metric for evaluating workflow designs, but other definitions are likely to have more desirable properties.

Proposition 13. *Given a set of roles $\mathcal{R}$, a responsibility for each role $\varphi_{\mathcal{R}}$, and a set of utility goals, there exists a graph-based workflow $(\varphi, \varphi_{\mathcal{R}})$ that is minimal among graph-based workflows.*

A minimal graph-based workflow can be computed using brute force by iteratively increasing the tags permitted on each edge of the workflow graph and testing whether the workflow achieves utility.

5.3 Auditing Workflow Execution

In evaluating workflow designs, we consider only responsible executions of the workflow. In an actual deployment, agents can act irresponsibly, either out of malice or by mistake, leading to policy violations. To hold agents accountable for these actions, organizations should record communication in an *audit log*. In this section, we present auditing algorithms for finding agents accountable for policy violations and for periodically scanning the log for signs of irresponsible actions. These algorithms are not fully automatic, but require an oracle that reports the actual contents of messages. We seek to minimize the number of oracle calls because we expect the oracle to be implemented by a human auditor. Additionally, we recommend the audit log maintain the Lamport causality [43] relation between events to facilitate efficient auditing.

5.3.1 Policy Violations and Accountability

Whenever a safety property is violated, it is violated on a finite trace, but the agent who performed the last action in that trace might not be blameworthy. For example, HIPAA does not permit the publication of protected health information in a newspaper, but HIPAA does not hold the reporter accountable for publishing the information. Instead, the person in the hospital who leaked the information caused the violation by acting irresponsibly and should be held accountable. Below, we make this intuition precise by defining *policy violation*, *causality*, and *accountability*.

Policy Violations. To define when an action violates a policy, we employ the notion of strong compliance [14]: an action is strongly compliant with a policy if there exists a continued execution that satisfies the policy. Formally, given a finite past history σ, an action a *strongly complies* with a policy θ, written $a \in \text{compliant}_\theta(\sigma)$, if there exists a trace σ' such that $\sigma \cdot a \cdot \sigma' \models \theta$. We require of policies that agents can determine whether their actions strongly comply with the policy. This ensures that policy violations are visible to the agents violating the policy and prevents policy compliance from depending on unrelated actions.

Def. A privacy policy θ has *local compliance* if, for all agents p and all traces π_1, π_2,

$$\pi_1 \restriction p = \pi_2 \restriction p \text{ implies } \text{compliant}_\theta(\pi_1) \cap A_p = \text{compliant}_\theta(\pi_2) \cap A_p,$$

where A_p is the set of actions for which p is the sender.

Causality. Workflows do not define any explicit causal relation between messages. For this reason, we resort to a standard trace-based notion of causality [43].

Def. The *possibly-caused* relation for a trace π, written $\rightsquigarrow_\pi$, is the minimal transitive relation such that $i \rightsquigarrow_\pi j$ if event i occurs before event j in the view $\pi \restriction p$ of some agent p.

In a trace π, the set of *causes* of an event j is the set of events $\text{causes}_\pi(j) = \{i \mid i \rightsquigarrow_\pi j\}$. The set of causes of an event is an over-approximation in the sense that if an event i actually caused event j (under some non-trace-based notion of actual causation, such as [45]), then $i \in \text{causes}_\pi(j)$, but it is possible that $i \in \text{causes}_\pi(j)$ without i being an actual cause for j.

Accountability. An agent is *accountable* for policy violation i in a trace π if the agent undertook an action in $\text{causes}_\pi(i)$ and did not fulfill his or her responsibilities in π. This definition is also an over-approximation because every agent whose irresponsibility actually caused a policy violation is classified as accountable, but not every accountable agent actually caused a policy violation.

Lemma 14 (Accountability)**.** *For all policies with local compliance, all graph-based workflows achieving the privacy policy, and all traces π, if π contains an action that violates the policy, then there exists an accountable agent.*

Proof. Given a trace π with an action i undertaken by agent p that violates the privacy policy, we construct a trace $\hat{\pi}$ that contains only the causes of i. The events $\hat{\pi}$ form a trace because the actions available to each agent at a given time depend only on the set of messages previously received by that agent, and all those events are included in $\hat{\pi}$. Moreover, $\pi \restriction p = \hat{\pi} \restriction p$ because all the events in p's view possibly

caused event i. Because the privacy policy has local compliance, i is also a violating event in $\hat{\pi}$, and there must exist an irresponsible agent q who undertook an action that possibly caused the violation. Finally, q's actions are also irresponsible in π and q is an accountable agent. □

5.3.2 Finding Accountable Agents

Our auditing algorithm for finding accountable agents in an audit log uses an oracle $\mathcal{O}$ such that $\mathcal{O}(m) = \text{contents}(m)$ to determine whether an agent acted responsibly by comparing the actual contents of messages with their tags. In practice, this oracle can be implemented by a human auditor, possibly with the assistance of a text classification algorithm.[2] The algorithm is formulated as a search on the causality graph of the audit log. The *causality graph* of a trace π is the graph with a node for each event in π and with an edge from event i to event j iff (1) $i \rightsquigarrow_\pi j$ and (2) there is no event k with $i \rightsquigarrow_\pi k$ and $k \rightsquigarrow_\pi j$. The causality graph can be constructed mechanically because it does not depend on the contents of the messages. Moreover, it can be constructed incrementally because actions are logged.

Algorithm 15. *Given a policy violation i and an audit log π, let G be the causality graph of π with the edges reversed. Beginning at i, perform a breath-first search of G. Upon encountering an action* $\mathsf{Send}(p, q, m)$*, compare* $\mathcal{O}(m)$ *with* $\text{tags}(m)$*. If p failed to add a tag for which he or she was responsible, output p as an accountable agent and terminate.*

If the human auditor decides that the agent found by the algorithm did not actually cause the policy violation, he or she can continue the search. The algorithm will eventually enumerate all the accountable agents, one of whom must have actually caused the policy violation.

[2]The University of Medicine & Dentistry of New Jersey determines whether to encrypt outgoing messages by scanning the messages for keywords [68], essentially guessing the correct tags mechanically.

Theorem 16 (Correctness)**.** *For every violation of a policy with local compliance in an audit log of a graph-based workflow achieving the policy, the algorithm outputs an accountable agent.*

Proof. Lemma 14 guarantees the existence of an irresponsible agent who possibly caused the policy violation. In graph-based workflows, irresponsible agents can be recognized by comparing the actual contents of messages with the tags of the messages. The algorithm searches the events that possibly caused the violation for such irresponsible tagging of messages. □

The algorithm reduces the number of oracle calls by restricting the search to actions that possibly caused the policy violation. In a deployment with thousands of agents exchanging messages, the portion of the log examined is likely to be several orders of magnitude smaller than the entire log, especially if an accountable agent is found at a shallow depth in the causality graph. The algorithm can use fewer oracle calls (while maintaining correctness) if the human auditor guides the search towards events that appear to be more relevant to the policy violation. The search can also be directed towards suspicious actions, a mechanical heuristic developed in the following section.

5.3.3 Monitoring for Irresponsible Actions

In addition to finding accountable agents after a policy violation occurs, auditing can also prevent some policy violations by detecting irresponsible actions before they lead to violations. The workflow engine can prevent irresponsible actions that can be detected mechanically, but it cannot prevent all irresponsible actions. In this section, we consider the problem of detecting irresponsible actions with the help of the oracle, but while minimizing work done by the human auditor. The simplest approach to detecting irresponsible actions in graph-based workflows is to sample communication at random and check, using the oracle, whether the sending agent tagged the message responsibly. This simple auditing algorithm requires many oracle calls to find a few irresponsible actions if the vast majority of communications are tagged responsibly. We suggest a more sophisticated approach based on a heuristic for *suspicious* actions.

Suspicious Events. Events appear suspicious if they indicate that some message tags were incorrect. The auditing engine can track the knowledge state of each agent as in Section 3.2.2, using the tags as a proxy for message contents:

$$\kappa \xrightarrow{\mathsf{Send}(p,q,m)} \kappa[q \mapsto \mathrm{cl}_{\preceq}(\kappa_q \cup \mathrm{tags}(m))]$$

The set of available actions can also be modified to use tags:

$$\mathrm{available}_p^{\mathrm{tags}}(\kappa) = \{\tau\} \cup \{\mathsf{Send}(p, q, m) \mid \mathrm{tags}(m) \subseteq \kappa_p\}$$

These definitions are feasible for mechanical agents because they rely on message tags instead of message contents. If messages were always tagged correctly, the knowledge states computed from the tags would coincide with the real knowledge states and every action undertaken by an agent would appear available from the tags. If some of the tags are incorrect, however, the apparent knowledge states are unlikely to coincide with the actual knowledge of agents. When agents undertake apparently unavailable actions, those actions are suspicious.

Def. An action a by agent p is *suspicious* if $a \notin \mathrm{available}_p^{\mathrm{tags}}(\kappa)$, where κ is the current knowledge state computed using message tags.

"Suspicious" is a heuristic. Irresponsible actions can lead to policy violations without triggering suspicion, and suspicious actions can occur unrelated to any irresponsible action. Suspicious actions do indicate related incorrect tags.

Proposition 17. *For all audit logs π, if agent p undertakes a suspicious action, then there exists a message in $\pi \restriction p$ with incorrect tags.*

Proof. The actions available to an agent are determined by his or her view of a trace. If an agent undertakes a suspicious action, then his actual knowledge differs from that computed from tags and thus a tag must be incorrect. □

Algorithm 18. *Given a suspicious event i by agent p and an audit log π, consult the oracle $\mathcal{O}$ about the message sent during i and the messages received by p before*

i in reverse chronological order until an incorrectly tagged message is found. If the message was tagged irresponsibly, output the sending agent.

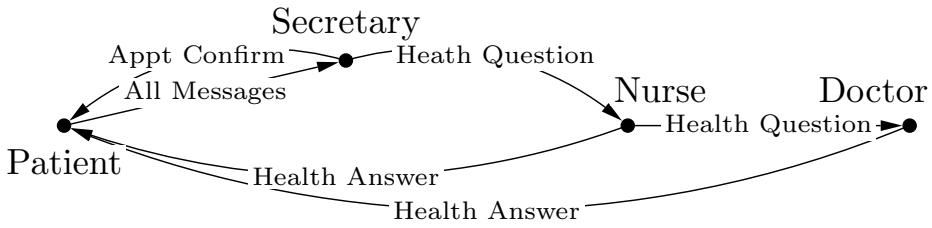
Chapter 6

Case Study: MyHealth@Vanderbilt

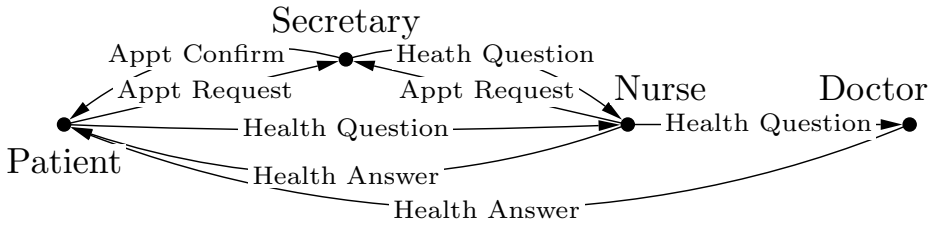
In this chapter, we illustrate the formal concepts developed in the preceding chapters using the MyHealth@Vanderbilt patient portal. Based on this formal analysis, we make several concrete design recommendations to the MyHealth developers. Specifically, we recommend introducing tags into the system and modifying the workflow to route health questions directly to nurses. Portions of this chapter appear in [15].

6.1 Overview

The MyHealth patient portal at Vanderbilt University Hospital allows patients to interact with their doctors and other healthcare professionals through a web-based messaging system (see `https://www.myhealthatvanderbilt.com/`). This innovative system at a leading research hospital, like related commercial ventures elsewhere, aims to provide better medical care at reduced costs, in a way that is more convenient to patients. In the MyHealth system, patients ask health questions and receive answers by exchanging messages with their doctors, labs send test results to patients, and patients send appointment requests to secretaries. Although HIPAA forbids some email communication with patients, the portal's patient authentication mechanism lets web-based messaging systems comply with HIPAA. The graph below depicts a portion of the MyHealth message passing system that deals with appointment requests and health questions.



This portion of MyHealth serves two utility goals: patients can schedule appointments and can receive answers to health questions. All messages are directed to the secretary, who is responsible for scheduling appointments and forwarding health questions to the nurse. This design reduces the doctor's workload, but the secretary learns the patient's sensitive health information contained in health questions. A more privacy conscious MyHealth design would deliver health questions directly to the nurse, bypassing the secretary. However, patients write messages in free text, making it difficult for MyHealth to route health questions to nurses and appointment requests to secretaries. We suggest permitting patients to tag messages with their contents using a simple drop-down control on the portal, enabling MyHealth to route message according to their tag. The graph below depicts our proposed tag-based workflow for MyHealth.



Our initial workflow proposal did not permit communication between the secretary and the nurse, but that workflow failed to achieve a utility goal. Vanderbilt employees would have been unable to answer health questions if a patient mistakenly tagged a health question as an appointment request as the secretary would be unable to correct the tag and forward the question to the nurse.

6.2 Workflow

6.2.1 Roles and Attributes

MyHealth is a graph-based workflow with roles *patient*, *secretary*, *nurse*, and *doctor* and attributes *appt-request*, *appt-confirm*, *health-question*, and *health-answer*.

$$\mathcal{R} = \{patient, secretary, nurse, doctor\}$$
$$\mathcal{T} = \{appt\text{-}request, appt\text{-}confirm, health\text{-}question, health\text{-}answer\}$$

There are two subsumption relations between attributes, enabling health answers to be computed from health questions and appointment confirmations to be computed from appointment requests.

$$health\text{-}answer \preceq health\text{-}question \qquad appt\text{-}confirm \preceq appt\text{-}request$$

In the initial knowledge state, patients know their own *health-question* and *appt-request*, and no other attributes are known.

6.2.2 Graph

The edges of the workflow graph are labeled as depicted in Section 6.1. For example, the edges emanating from the nurse to the other roles are labeled as follows:

$$T_{secretary,patient} = \{appt\text{-}confirm\} \qquad T_{patient,secretary} = \{appt\text{-}request\}$$
$$T_{nurse,patient} = \{health\text{-}answer\} \qquad T_{nurse,secretary} = \{appt\text{-}request\}$$
$$T_{doctor,patient} = \{health\text{-}answer\} \qquad T_{doctor,secretary} = \emptyset$$
$$T_{patient,nurse} = \{health\text{-}question\} \qquad T_{patient,doctor} = \emptyset$$
$$T_{secretary,nurse} = \{health\text{-}question\} \qquad T_{secretary,doctor} = \{appt\text{-}confirm\}$$
$$T_{doctor,nurse} = \emptyset \qquad T_{nurse,doctor} = \{health\text{-}question\}$$

6.2.3 Responsibilities

MyHealth has several progress responsibilities.

Secretary. Secretaries are responsible for confirming appointments and for forwarding health questions to nurses. Note that although the workflow graph does not deliver to secretaries messages tagged as containing health questions, secretaries might receive health questions that are mistagged as containing appointment requests.

$$\mathbf{G}\forall p, q, m.\, \mathsf{send}(p, x, m) \wedge \mathsf{contains}(m, q, \textit{appt-request}) \rightarrow \mathbf{F}\exists m'.\, \mathsf{contains}(m', q, \textit{appt-confirm}) \wedge \mathsf{send}(x, q, m')$$

$$\mathbf{G}\forall p, q, m.\, \mathsf{send}(p, x, m) \wedge \mathsf{contains}(m, q, \textit{health-question}) \rightarrow \mathbf{F}\exists n.\, \mathsf{inrole}(n, \textit{nurse}) \wedge \mathsf{send}(x, n, m)$$

Nurse. Nurses are responsible for either answering or forwarding health questions, for sending misdirected appointment requests to the secretary, and for sending health answers only to the appropriate patient.

$$\mathbf{G}\forall p_1, q, m.\, \mathsf{send}(p_1, x, m) \wedge \mathsf{contains}(m, q, \textit{health-question}) \rightarrow \mathbf{F}(\exists d.\, \mathsf{inrole}(d, \textit{doctor}) \wedge \mathsf{send}(x, d, m)) \vee (\exists m'.\, \mathsf{contains}(m', q, \textit{health-answer}) \wedge \mathsf{send}(x, q, m'))$$

$$\mathbf{G}\forall p, q, m.\, \mathsf{send}(p, x, m) \wedge \mathsf{contains}(m, q, \textit{appt-request}) \rightarrow \mathbf{F}\exists s.\, \mathsf{inrole}(s, \textit{secretary}) \wedge \mathsf{send}(x, s, m)$$

$$\mathbf{G}p, q, m.\, \mathsf{send}(x, p, m) \wedge \mathsf{contains}(m, q, \textit{health-answer}) \rightarrow p = q$$

Doctor. Doctors are responsible for answering health questions they receive.

$$\mathbf{G}\forall p, q, m.\, \mathsf{send}(p, x, m) \wedge \mathsf{contains}(m, q, \textit{health-question}) \rightarrow \mathbf{F}\exists m'.\, \mathsf{contains}(m, q, \textit{health-answer}) \wedge \mathsf{send}(x, q, m')$$

$$\mathbf{G}p, q, m.\, \mathsf{send}(x, p, m) \wedge (\mathsf{contains}(m, q, \textit{health-question}) \vee \mathsf{contains}(m, q, \textit{health-answer})) \rightarrow p = q$$

6.3 Evaluation

Privacy. The salient privacy goal of MyHealth is to comply with HIPAA. Many of the requirements of the HIPAA Privacy Rule can be expressed in the logic. For simplicity, we consider two specific requirements: only health care providers receive protected health information and patients receive only their own health answers.

$$\mathbf{G}\forall p_1, p_2, q, m.\, \mathsf{send}(p_1, p_2, m) \wedge \mathsf{contains}(q, \textit{health-question}) \rightarrow \mathsf{inrole}(p_2, \textit{nurse}) \vee \mathsf{inrole}(p_2, \textit{doctor})$$

$$\mathbf{G}\forall p_1, p_2, q, m.\, \mathsf{send}(p_1, p_2, m) \wedge \mathsf{inrole}(p_2, \textit{patient}) \wedge \mathsf{contains}(m, q, \textit{health-answer}) \rightarrow q = p_2$$

The proposed MyHealth workflow achieves both of these privacy goals, but the current workflow does not achieve the first. Although these properties are easy to check for this simple workflow, we could have used the algorithm in Section 5.2.1.

Utility. Two utility goals of MyHealth are that patients can schedule appointments and receive answers to their health questions. We state these goals in terms of the existence of strategies.

$$\forall p.\, \mathsf{inrole}(p, \mathit{patient}) \rightarrow \langle\langle p \rangle\rangle \mathbf{F} \exists q, m.\, \mathsf{send}(q, p, m) \wedge \mathsf{contains}(m, p, \mathit{health\text{-}answer})$$

$$\forall p.\, \mathsf{inrole}(p, \mathit{patient}) \rightarrow \langle\langle p \rangle\rangle \mathbf{F} \exists q, m.\, \mathsf{send}(q, p, m) \wedge \mathsf{contains}(m, p, \mathit{appt\text{-}confirm})$$

MyHealth does achieve these utility goals. Another utility goal is worth analyzing because some of the patients who use MyHealth might not be technically savvy. The nurse and doctor have a strategy for reacting to health questions sent by patients with health answers, even if the patient incorrectly tags his or her health question and it is directed to the secretary. MyHealth without patient responsibilities achieves this privacy goal, but a variant of the workflow that does not include edges between the secretary and the nurse does not achieve this utility goal.

$$\begin{aligned}
\forall s, n, d.\, \mathsf{inrole}(s, \mathit{secretary}) \wedge \mathsf{inrole}(n, \mathit{nurse}) \wedge \mathsf{inrole}(d, \mathit{doctor}) \rightarrow \\
\langle\langle s, n, d \rangle\rangle \mathbf{G} \forall p, x, m.\, \mathsf{inrole}(p, \mathit{patient}) \wedge ((x = s) \vee (x = n) \vee (x = d)) \wedge \\
\mathsf{send}(p, x, m) \wedge \mathsf{contains}(m, p, \mathit{health\text{-}question}) \rightarrow \\
\mathbf{F} \exists m'.\, \mathsf{contains}(m', p, \mathit{health\text{-}answer}) \wedge \\
(\mathsf{send}(n, p, m') \vee (\mathsf{send}(d, p, m'))
\end{aligned}$$

Auditing. We illustrate the auditing algorithms on an example trace π that violates MyHealth's privacy goals. Suppose Alice, a secretary, received a health answer intended for a patient. Alice received the message because it was sent by Bob, another secretary, and tagged as an appointment request. Tracing backwards through the causality structure, the auditing algorithm queries the human auditor for the contents of several messages sent to Bob until it find an irresponsible message sent by Carol, a nurse. Carol sent the health answer to Bob irresponsibly tagged as an appointment request and can be held accountable for this action. Moreover, this irresponsible action is likely to be found by monitoring because it is suspicious: Bob had not previously received a message tagged as an appointment request.

Chapter 7

Conclusion

Privacy and regulatory compliance are important business and social concerns. Existing laws and societal expectations are complex and difficult for modern enterprises to understand and manage. We believe there is a need for a general, clear, and comprehensive framework for reducing high-level requirements to specific operating guidelines that can be applied at individual steps in a business process or organizational workflow.

Privacy policies concern the use and disclosure of personal information. An individual's personal information is structured in that knowledge of some attributes, such as postal address, contain information about other attributes, such as postal code. In the extreme case, an agent who knows the value of one attribute can exactly determine the value of another attribute. This relation between attributes, known as the data hierarchy, complicates privacy languages because statements about one attribute have consequences for the use and disclosure of other attributes.

Existing policy languages, such as P3P and EPAL, contain semantic anomalies arising from the interrelation of personal attributes. In P3P, these anomalies take the form of not guaranteeing that a consumer's privacy preferences, expressed in either APPEL or XPref, will be respected if a service provider lives up to its P3P policy. Preference authors can work around these anomalies by using only monotonic preferences, i.e., those preferences that are free of negation.

EPAL also contains semantic anomalies related to the data hierarchy. These anomalies give rise to unsafe policies that impose fewer restrictions on more generic actions. These policies are anomalous because they let a member of the enterprise avoid policy obligations by phrasing his or her policy query in more general terms. Complications introduced by the data hierarchy also prevent EPAL from being closed under combination. Given two EPAL policies, there might not exist a third EPAL policy that enforces their conjunction.

Although these semantic anomalies can be repaired via involved lattice constructions [16], the complexities arising from the data hierarchy can be more cleanly dealt with by improving the foundations of privacy languages. Instead of reasoning about the inheritance of policy statements (as in [12]), we suggest reasoning about the deductive power of agents. We model the knowledge state of agents and then update their knowledge as state they receive messages from other agents. By requiring that the messages exchanged between agents are downwardly closed under the data hierarchy relation, we are able to cleanly avoid the semantic anomalies that plague other privacy languages.

Our model is inspired by contextual integrity, a conceptual framework for understanding privacy expectations that has been developed in the literature on law and public policy. Contextual integrity holds that whether a communication is appropriate (meets societal expectations about privacy) depends on the context, the role, and the subject of personal information, and cannot be captured accurately using a DRM-style "ownership of information" model or a simple partitioning of information into "public information" and "private information."

By articulating these features in the model, the logic of privacy and utility we develop formalizes the central ideas of contextual integrity in a well-studied temporal logic. Privacy norms are expressed in the Linear-time Temporal Logic (LTL) fragment, which is interpreted over traces of basic communication actions of the form "Alice gives Bob a particular type of information about Carol." The temporal operators in the logic correspond to "principles of transmission" in contextual integrity. For example, if Alice sends information to Bob under the principle of confidentiality, then, in the future, Bob must refrain from sending that information to a third party.

The temporal operators are also useful for capturing provisions in privacy policies that refer to the past (for example, that the subject opted in to a particular kind of communication) or to the future (for example, that the subject must be notified that the communication occurred).

It is difficult to evaluate a privacy language (or, more generally, a policy language) scientifically. To evaluate the privacy fragment of the logic of privacy and utility, we present algorithms for checking policy consistency, combining policies, and enforcing compliance. Unlike in previous privacy languages, these algorithmic problems reduce to well-studied problems in temporal logic, letting us leverage a large body of prior work. Policy consistency amounts to satisfiability. Policy combination, which is problematic in EPAL, is formulated easily using logical conjunction and disjunction. Enforcing compliance is subtle, distilling into two notions: weak and strong compliance. Weak compliance is computable in polynomial time per action but guarantees only that an action meets all present requirements (future obligations might be impossible to satisfy). Strong compliance requires polynomial space but guarantees that the agents can discharge their future requirements. In most policies we have examined, weak compliance is sufficient because the future requirements entailed by the policy are always possible to discharge.

We also evaluate our privacy language by comparing its expressiveness with that of existing privacy and policy languages. Specifically, we compare expressiveness with RBAC, XACML, EPAL, and P3P. Our results are summarized in Fig. 4.1. Our language is roughly a superset of these other languages, with a number of subtle exceptions. For example, XACML lets policies invoke arbitrary programs during policy evaluation, engendering expressiveness not available in our language. Our temporal conditions improve on the uninterpreted future obligations of XACML and EPAL by letting our algorithms for policy combination and compliance reason directly about these obligations.

Finally, we evaluate the privacy language by trying to express privacy provisions of federal regulations including HIPAA, COPPA, and GLBA. We find that we are able to express the main privacy provisions of these regulations and that the regulations exercise all the features of our language. Some regulations, such as HIPAA, consist

mostly of positive norms, enumerating the allowed communications and forbidding all other communication. Other regulations, such as COPPA and GLBA, consist mostly of negative norms, forbidding some kinds of communication or imposing obligations whenever other kinds of communication occur. The different senses of these requirements are captured accurately in the logic.

The formalized regulations in Section 4.3 are useful for evaluating the expressiveness of the logic of privacy and utility, but they are not complete enough to be used to enforce compliance with the regulation. Creating a complete formalization of the privacy provisions of a regulation is a significant undertaking. We have made some amount of progress formalizing the HIPAA Privacy Rule in a Horn fragment of the logic [65]. One significant challenge in this work is providing assurance that our translation accurately captures the semantics of the regulation.

There are a number of directions for expanding the privacy language by extending the logic. For example, our model assumes that policies are based only on the type of information (rather than actual data values) and that information is about a single individual (rather than about a group of individuals). We could extend the formalization by relaxing these restrictions, letting policies depend on specific data values and letting information describe groups of individuals.

Our current language faces a limitation common to many policy languages. Consider SB 1386, a California law requiring businesses that inappropriately disclose personal information to notify the subjects of the information. This provision cannot be expressed properly in the language because it takes effect only when an agent violates a policy. In our model, an agent either does or does not comply with a privacy policy. In either case, the agent need not send notifications. However, Californians receive such notifications regularly. To express such "defense in depth" provisions, we could extend our logic with a modality as in deontic logic [**?**]. Unfortunately, the known deontic logics suffer semantic anomalies with such "contrary-to-duty" imperatives [**?**].

One extension that we have investigated in detail is extending the privacy language to include notions of utility. Utility differs from privacy in that utility is not a trace-based property. A given workflow or business process is useful in virtue of its *possible* executions. For example, a patient health portal is useful for answering the health

questions of patients if those patients can get their health questions answered—even if no patients actually ask any questions in a given trace. To accurately capture this notion of utility, we expand the logic beyond simple linear-time to Alternating-time Temporal Logic (ATL). The ATL strategy quantifier lets us reason about what an agent (or a coalition of agents) can accomplish in a workflow.

Our unified model of agents interacting via a concurrent game structure to execute a workflow lets us examine questions that concern both privacy and utility. For example, this model is expressive enough to capture privacy requirements such as "minimal necessary," which refers to maximizing privacy while maintaining some fixed amount of utility. The authors of these provisions seem to view privacy and utility as directly in tension: tightening restrictions on data disclosure enhances privacy at the expense of utility. This view is not substantiated in our model. In fact, there are examples in which both privacy and utility are enhanced by adding more restrictions on information disclosure. These situations occur when restricting the actions of one agent creates utility for another agent because that agent now has a strategy for bringing about some useful end.

We also distinguish between information visible to mechanical agents and to human agents. Specifically, we assume messages have explicit tags that can be read by mechanical agents, but we do not assume that these tags match the actual semantic contents of messages. This distinction is useful in separating those enforcement tasks that can be carried out by electronic systems from those tasks that require human agents. We assume that electronic agents are online and can monitor and restrict communication to the best of their abilities but that human enforcement agents participate only after-the-fact, as auditors. By structuring the model in this way, we can examine algorithms for design-time privacy and utility analysis of workflows as well as algorithms for run-time auditing of workflow execution.

The design-time privacy analysis applies the general algorithms for our privacy language to the specific case of comparing a privacy policy with a workflow design, as represented by a set of responsibilities for each agent in the workflow. We say that a workflow achieves its privacy goals if the agents satisfy the privacy policy when they all live up to their responsibilities. The design-time utility analysis requires a sound

approximation algorithm to overcome an undecidability result about ATL* model checking with imperfect information. We then use this algorithm to evaluate whether responsible agents satisfy the utility goals of the workflow.

At run-time, agents might not fulfill their responsibilities, leading to privacy violations. One strategy for dealing with these breaches is to perform regular audits and hold individuals accountable for their actions. To be held accountable for a privacy breach, however, an agent not only must have been derelict in his or her responsibilities but also must have *caused* the privacy violation. Conversely, simply having caused a privacy breach is not, in and of itself, enough grounds for holding an agent accountable for the breach. To be held accountable, that agent must also have been responsible for preventing the breach.

To make precise the notion of causation, we appeal to the classic definition of Lamport causality in distributed systems. Using this definition, we develop auditing algorithms for using an audit log to find an agent to hold accountable for a privacy violation. These algorithms make use of a human auditor to uncover the semantic contents of messages. We evaluate the efficiency of these algorithms by asking that the algorithms economize their use of the human auditor, modeled as an oracle. These algorithms are practical and efficient but unlikely to be the final word on the auditing problem. We plan to investigate this problem further in subsequent work, specifically exploring fully automated auditing techniques with reasonable false positive and false negative rates and developing (semi-)automated techniques for identifying policy violations by analyzing audit logs.

We apply these methods to analyze the design of MyHealth@Vanderbilt, a patient portal deployed at Vanderbilt Medical Center. Our analysis leads to concrete suggestions for improving the privacy and utility of the workflow. We believe that the methods in this thesis are applicable to systems in a wide range of sectors including health care and financial services, where business processes routinely handle personal information and privacy and utility concerns are significant. In subsequent work, we hope to carry out case studies of other health care systems using automated tools. Another application area we hope to investigate is the outsourcing of business processes that deal with sensitive information such as social security and credit card

numbers. In this setting, minimal workflows are particularly useful for tasks such as credit card charge-back that require access to real credit card numbers.

While it is unreasonable to expect the manager of a hospital, call center, or credit card processing organization to become fluent in temporal logic, we believe that the most productive way to address the basic problem is to develop precise, unambiguous foundations and use them to develop more accessible principles and guidelines.

Bibliography

[1] Mark S. Ackerman, Lorrie Faith Cranor, and Joseph Reagle. Privacy in e-commerce: Examining user scenarios and privacy preferences. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 1–8. ACM Press, 1999.

[2] Nabil R. Adam, Vijayalakshmi Atluri, and Wei-Kuang Huang. Modeling and analysis of workflows using petri nets. *J. Intell. Inf. Syst.*, 10(2):131–158, 1998.

[3] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. An XPath-based preference language for P3P. In *Proceedings of the Twelfth International Conference on World Wide Web*, pages 629–639. ACM Press, 2003.

[4] Rakesh Agrawal, Ramakrishnan Srikant, and Dilys Thomas. Privacy preserving OLAP. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 251–262, New York, NY, USA, 2005. ACM Press.

[5] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[6] Anne Anderson. Key differences between XACML and EPAL. Ottawa new challenges for access control, 2005.

[7] Anne Anderson, Anthony Nadalin, Bill Parducci, Daniel Engovatov, Ed Coyne, Frank Siebenlist, Hal Lockhart, Michael McIntosh, Michiharu Kudo, Polar Humenn, Ron Jacobson, Seth Proctor, Simon Godik, Steve Anderson, and Tim Moses. Extensible access control markup language (XACML) version 2.0, 2004.

[8] Annie I. Antón, Julia Brande Earp, and Angela Reese. Analyzing website privacy requirements using a privacy goal taxonomy. In *Requirements Engineering 2002*, pages 23–31, 2002.

[9] Annie I. Antón, Qingfeng He, and David L. Baumer. Inside JetBlue's privacy policy violations. *IEEE Security and Privacy*, 2(6):12–18, 2004.

[10] Vijayalakshmi Atluri and Wei-Kuang Huang. An authorization model for workflows. In *European Symposium on Research in Computer Security (ESORICS)*, volume 1146 of *LNCS*, pages 44–64. Springer–Verlag, 1996.

[11] M. Backes, B. Pfitzmann, and M. Schunter. A toolkit for managing enterprise privacy policies. In *European Symposium on Research in Computer Security (ESORICS)*, volume 2808 of *LNCS*, pages 101–119. Springer–Verlag, 2003.

[12] Michael Backes, Markus Dürmuth, and Rainer Steinwandt. An algebra for composing enterprise privacy policies. In *European Symposium on Research in Computer Security (ESORICS)*, volume 3193 of *LNCS*. Springer–Verlag, 2004.

[13] Michael Backes, Günter Karjoth, Walid Bagga, and Matthias Schunter. Efficient comparison of enterprise privacy policies. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, pages 375–382. ACM Press, 2004.

[14] Adam Barth, Anupam Datta, John C. Mitchell, and Helen Nissenbaum. Privacy and contextual integrity: Framework and applications. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 184–198, Washington, DC, USA, 2006. IEEE Computer Society.

[15] Adam Barth, Anupam Datta, John C. Mitchell, and Sharada Sundaram. Privacy and utility in business processes. In *IEEE Computer Security Foundations Symposium*, pages 279–294. IEEE Computer Society, July 2007.

[16] Adam Barth and John C. Mitchell. Enterprise privacy promises and enforcement. In *Workshop on Issues in the Theory of Security*, pages 58–66. ACM Press, 2005.

[17] Adam Barth, John C. Mitchell, and Justin Rosenstein. Conflict and combination in privacy policy languages. In *Proceedings of the 2004 Workshop on Privacy in the Electronic Society*. ACM Press, 2004.

[18] Moritz Y. Becker and Peter Sewell. Cassandra: Flexible trust management, applied to electronic health records. In *IEEE Computer Security Foundations Workshop*. IEEE Computer Society, 2004.

[19] D. E. Bell and L. J. LaPadula. Secure computer systems: Mathematical foundations and model. Technical Report M74-244, Mitre Corp, 1975.

[20] Matt Bishop. *Computer Security: Art and Science*. Addison Wesley Professional, 2003.

[21] Simon Byers, Lorrie Faith Cranor, and David Kormann. Automated analysis of P3P-enabled web sites. In *Proceedings of the 5th International Conference on Electronic Commerce*, pages 326–338. ACM Press, 2003.

[22] Shuchi Chawla, Cynthia Dwork, Frank McSherry, Adam Smith, and Hoeteck Wee. Toward privacy in public databases. In *TCC '5: Proceedings of the Second Theory of Cryptography Conference*, volume 3378 of *LNCS*, pages 363–385. Springer–Verlag, 2005.

[23] James Clark and Steve DeRose. XML path language (XPath), 1999. `http://www.w3.org/TR/xpath`.

[24] CNN. FBI seeks stolen personal data on 26 million vets. `http://www.cnn.com/2006/US/05/22/vets.data/`.

[25] Jason Crampton. On permissions, inheritance and role hierarchies. In *Proceedings of the 10th ACM Conference on Computer and Communication Security*, pages 85–92. ACM Press, 2003.

[26] Lorrie Faith Cranor. *Web Privacy with P3P*. O'Reilly and Associates, Inc., 2002.

[27] Lorrie Faith Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, and Joseph Reagle. The platform for privacy preferences 1.0 (P3P1.0) specification. `http://www.w3.org/TR/P3P/`, 2002.

[28] CSO Online. Safety in numbers. `http://www.csoonline.com/metrics/viewmetric.cfm?id=265`.

[29] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *POLICY '01: Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, pages 18–38, London, UK, 2001. Springer–Verlag.

[30] Stéphane Demri and Philippe Schnoebelen. The complexity of propositional linear temporal logics in simple cases. In *Procceding of the 15th Annual Symposium on Theoretical Aspects of Computer Science (STACS'98)*, volume 1373 of *LNCS*. Springer–Verlag, 1998.

[31] John DeTreville. Binder, a logic-based security language. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 105, Washington, DC, USA, 2002. IEEE Computer Society.

[32] Marlon Dumas and Arthur H. M. ter Hofstede. UML activity diagrams as a workflow specification language. In *UML 2001*, pages 76–90. Springer–Verlag, 2001.

[33] Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO 2004: 24th Annual International Cryptology Conference*, volume 3152 of *LNCS*, pages 528–544. Springer–Verlag, 2004.

[34] EPIC. The Gramm–Leach–Bliley Act. `http://epic.org/privacy/glba/`.

[35] Federal Trade Commission. How to comply with the children's online privacy protection rule. `http://www.ftc.gov/bcp/conline/pubs/buspubs/coppa.htm`, 1999.

[36] Federal Trade Commission. In brief: the financial privacy requirements of the Gramm–Leach–Bliley Act. `http://www.ftc.gov/bcp/conline/pubs/buspubs/glbshort.htm`, 2002.

[37] Gerard J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison Wesley Professional, 2004.

[38] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, 2001.

[39] Wojciech Jamroga, Wiebe van der Hoek, and Michael Wooldridge. On obligations and abilities. In *Deontic Logic: 7th International Workshop on Deontic Logic in Computer Science*, volume 3065 of *LNCS*, pages 165–181. Springer–Verlag, 2004.

[40] Carlos Jensen and Colin Potts. Privacy policies as decision-making tools: An evaluation of online privacy notices. In *Proceedings of the 2004 Conference on Human Factors in Computing Systems*, pages 471–478. ACM Press, 2004.

[41] Günter Karjoth and Matthias Schunter. A privacy policy model for enterprises. In *IEEE Computer Security Foundations Workshop*, page 271. IEEE Computer Society, 2002.

[42] Krishnaram Kenthapadi, Nina Mishra, and Kobbi Nissim. Simulatable auditing. In *PODS '05: Proceedings of the 24th ACM Symposium on Principles of Database Systems*, pages 118–127, New York, NY, USA, 2005. ACM Press.

[43] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[44] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. DeWitt. Limiting disclosure in hippocratic databases. In *30th International Conference on Very Large Data Bases*, Toronto, Canada, August 2004.

[45] David Lewis. Causation as influence. *The Journal of Philosophy*, 97(4):181–197, 2000.

[46] Ninghui Li and John C. Mitchell. RT: A role-based trust-management framework. In *The Third DARPA Information Survivability Conference and Exposition*, pages 201–212, Washington, DC, USA, 2003. IEEE Computer Society.

[47] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-Diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

[48] Zohar Manna and Amir Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer–Verlag, 1995.

[49] Markle Foundation. The Connecting for Health Common Framework. `http://www.connectingforhealth.org/commonframework/`, 2006.

[50] Michael J. May, Carl A. Günter, and Insup Lee. Privacy apis: Access control techniques to analyze and verify legal privacy policies. In *IEEE Workshop on Computer Security Foundations*, pages 85–97. IEEE Computer Society, 2006.

[51] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *SP '08: Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 111–125, Washington, DC, USA, 2008. IEEE Computer Society.

[52] Helen Nissenbaum. Privacy as contextual integrity. *Washington Law Review*, 79(1):119–158, 2004.

[53] OASIS. OASIS Web Services Process Execution Language (WSBPEL). `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel`.

[54] Office for Civil Rights. Summary of the HIPAA privacy rule. US Department of Health & Human Services, 2003.

[55] Oracle. Oracle BPEL Process Manager. `http://www.oracle.com/technology/products/ias/bpel/`.

[56] J.E.J. Prins. The propertization of personal data and identities. *Electronic Journal of Comparative Law*, 8.3, October 2004.

[57] James Rachels. Why privacy is important. In Ferdinand David Schoeman, editor, *Philosophical Dimensions of Privacy: An Anthology*, pages 290–294. 1984.

[58] Joseph Reagle and Lorrie Faith Cranor. The platform for privacy preferences. *Communications of the ACM*, 42(2):48–55, 1999.

[59] Grigore Rosu and Klaus Havelund. Synthesizing dynamic programming algorithms for linear temporal logic formulae. Technical Report TR 01-15, RIACS, May 2001.

[60] Pierangela Samarati. Protecting respondent's privacy in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[61] Ferdinand Schoeman. Privacy and intimate information. In Ferdinand David Schoeman, editor, *Philosophical Dimensions of Privacy: An Anthology*, pages 403–408. 1984.

[62] Ferdinand Schoeman. Gossip and privacy. In Robert F. Goodman and Aaron Ben-Zeev, editors, *Good Gossip*, pages 403–408. 1994.

[63] Matthias Schunter, Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers, and Matthias Schunter. Enterprise privacy authorization language (EPAL 1.1). `http://www.zurich.ibm.com/security/enterprise-privacy/epal/Specification/`, 2003.

[64] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, July 1985.

[65] Sharada Sundaram, Anjali Behal, and Adam Barth. HIPAA compliance checker. `http://hipaa.googlecode.com/`, 2008.

[66] Latanya Sweeney. k-Anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[67] Wouter Teepe, Reind P. van de Riet, and Martin S. Olivier. Workflow analyzed for security and privacy in using databases. In *Working Conference on Database Security*, pages 271–282. Kluwer, B.V., 2001.

[68] University of Medicine & Dentistry of New Jersey. HIPAA Security Standards FAQ's. `http://www2.umdnj.edu/hipaaweb/security/security_emailFAQ02.htm`, 2007.

[69] Vanderbilt Medical Center. MyHealthAtVanderbilt. `https://www.myhealthatvanderbilt.com/`, 2007.