



AFRL-RI-RS-TR-2018-193

ORIGIN PRIVACY – PROTECTING PRIVACY IN THE BIG-DATA ERA

NEW YORK UNIVERSITY

AUGUST 2018

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2018-193 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /

CARL R. THOMAS
Work Unit Manager

/ S /

JOHN D. MATYJAS
Technical Advisor, Computing
& Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE (DD-MM-YYYY) AUG 2018		2. REPORT TYPE FINAL TECHNICAL REPORT		3. DATES COVERED (From - To) AUG 2016 – FEB 2018	
4. TITLE AND SUBTITLE ORIGIN PRIVACY – PROTECTING PRIVACY IN THE BIG-DATA ERA				5a. CONTRACT NUMBER N/A	
				5b. GRANT NUMBER FA8750-16-2-0287	
				5c. PROGRAM ELEMENT NUMBER 62303E	
6. AUTHOR(S) Helen Nissenbaum, Sebastian Benthall, Anupam Datta, Michael Carl Tschantz, Piot Mardziel				5d. PROJECT NUMBER BRAN	
				5e. TASK NUMBER NY	
				5f. WORK UNIT NUMBER UN	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) New York University, 70 Washington Sq. S., New York, NY 10012-1019; Carnegie Mellon University, 4720 Forbes Ave, Pittsburgh, PA 15213; International Computer Science Institute, 1947 Center St., Berkeley, CA 94704				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RITA 525 Brooks Road Rome NY 13441-4505 DARPA 675 North Randolph Street Arlington VA 22203-2114				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI	
				11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2018-193	
12. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Existing data protection laws sometimes restrict information flow based on its origin or provenance. These origin-based rules are distinct from information flow rules that restrict information based on its topic. In addition to sometimes being directly specified in law, information origin rules are less ambiguous than information topic rules. We call the conception of privacy as that maintained by complying with origin-based rules Origin Privacy. Our work motivates then formally specifies, implements, and analyzes origin rules to information flow.					
15. SUBJECT TERMS Contextual integrity, information flow security, causal Bayesian networks, noninterference, differential privacy, type system, policy specification					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 87	19a. NAME OF RESPONSIBLE PERSON CARL R. THOMAS
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code)

Origin Privacy

Protecting Privacy in the Big-Data Era

Contents

1.0 EXECUTIVE SUMMARY	3
2.0 INTRODUCTION	4
2.1 Prior Work	5
2.2 Structure	6
3.0 METHODS, ASSUMPTIONS, AND PROCEDURES	7
3.1 Theoretical frame	7
3.2 Concepts of Information Control	9
3.3 Contextual Integrity and Origin Privacy	11
3.4 Information flow and causal models	12
3.4.1 Philosophy: contextual integrity and information flow	12
3.4.2 Causal probabilistic graphical models	13
3.5 Ambiguity of information flow	16
4.0 RESULTS AND DISCUSSIONS	17
4.1 Policy motivations	17
4.1.1 Policy example: HIPAA	18
4.1.2 Policy example: GLBA	21
4.1.3 Policy example: PCI DSS	21
4.1.4 Other policies	22
4.2 Origin Privacy	23
4.2.1 Systems	23
4.2.2 Processes	24
4.2.3 Origin and provenance	24
4.2.4 Origin privacy	25
4.3 Syntax	25

4.4	General Semantics	26
4.4.1	Requests	26
4.4.2	Policy Semantics	28
4.5	Semantics of Origin	29
4.6	Implementation Prototype	31
4.6.1	Attributes as Labels	32
4.6.2	Policies	33
4.6.3	Conditions	34
4.6.4	Legalese-like Policies	34
4.6.5	Demo: Big Data Processing	35
4.7	Bayesian networks and information flow security	38
4.7.1	Embedded Causal System (ECS) Model	39
4.7.2	System design	40
4.7.3	Noninterference	42
4.7.4	Preventing associations	45
4.8	Formalizing origin privacy	46
4.9	Use case: IoT and biometric data	48
4.9.1	GDPR biometric data	48
4.9.2	Formalizing GDPR requirements	49
4.10	How to Author Policies for Origin Privacy	50
4.10.1	Overview	50
4.10.2	Modeling Contexts	51
4.10.3	Patient–Physician Example	52
4.10.4	Steps for Policy Selection and Enforcement	55
5.0	CONCLUSIONS	56
5.1	Specifiability Criterion	56
5.2	Observer capabilities	57
5.3	Incentives-based policy design	57
6.0	RECOMMENDATIONS	58
6.1	Declassification	58
6.2	Purpose	58
6.3	Relationship to Differential Privacy	58
6.3.1	Mutual information differential privacy	59
6.3.2	Randomizing database inputs	60
6.3.3	Generalizing ϵ -security	61
6.4	Incentives in ECS	62
6.4.1	Expert services model	63
6.4.2	Expert ECS Model	64
7.0	REFERENCES	67

A	DPI Theorems	72
A.1	Triplet Structures	72
A.2	Quartet Structures	75
A.3	Paths	76
B	Multi-Agent Influence Diagrams and Data Games	79
B.1	Multi-Agent Influence Diagrams	79
B.1.1	Tactical independence	81
B.1.2	Notation	81
B.2	Data Games	81
C	Gaps in Brandeis Presentations	82
D	Publications	84
E	Unpublished Reports	84

List of Figures

1	Example policy making use of data types	12
2	A policy example making use of origin in place of data types	12
3	Alice’s commute to work	14
4	Test score ranks (R) distributed to Bob (B) and Eve (E)	17
5	Example attribute and policy specification	26
6	A system that is safe so long as its high-side actuators are not observed	41
7	A system that is not propitious when E is unobserved	46

List of Tables

1	Outcomes from sharing and using information	51
---	---	----

1.0 EXECUTIVE SUMMARY

Existing data protection laws sometimes restrict information flow based on its origin or provenance. These origin-based rules are distinct from information flow rules that restrict information based on its topic. In addition to sometimes being directly specified in law, origin-based rules are less ambiguous than topic-based ones. In fact, it is under some conditions impossible to know that information does not allow inferences about some sensitive topic.

For these reasons, we explore basing privacy rules upon the origins of information. We call the conception of privacy as that maintained by complying with origin-based rules *Origin Privacy*. Our work motivates then formally specifies, implements, and analyzes origin-based rules to information flow.

The contributions of this work include:

- An analysis of privacy policies, specifically with respect to how they determine protected classes of information through information topic and information origin;
- An informal ontology and articulation of Origin Privacy appropriate for use by policy designers;
- Disambiguation of the concept of “information flow” into causal flow and nomic association components through causal modeling;
- A language for the specification of origin-based privacy rules;
- A type system for enforcing such policies;
- A prototype implementation of the type system;
- The Embedded Causal System (ECS) model, a model of a causal system embedded in its environment suitable for proving properties of information flow security under these conditions;
- Proofs of conditions for noninterference and semantic security in causal and embedded causal systems;
- Formal security models for origin noninterference and origin semantic security, with proofs of sufficient conditions;
- Demonstration of the use of Origin Privacy in a biometric Internet of Things use case;
- Relaxed security models based on mutual information and proofs of their formal relationship to differential privacy; and
- A demonstration of the use of ECS models in game theoretic modeling using Multi-Agent Influence Diagrams.

2.0 INTRODUCTION

Machine learning over big data poses challenges for our conceptualization of privacy. Such techniques can discover surprising and counteractive associations that take innocent looking data and turns it into important inferences about a person. For example, the buying carbon monoxide monitors has been linked to paying credit card bills, while buying chrome-skull car accessories predicts not doing so [1]. Also, Target may have used the buying of scent-free hand lotion and vitamins as a sign that the buyer is pregnant [2, 3]. If we take pregnancy status to be private and assume that we should prohibit the sharing information that can reveal that fact, then we have created an unworkable notion of privacy, one in which sharing any scrap of data may violate privacy.

Prior technical specifications of privacy depend on the classification of certain types of information as private or sensitive; privacy policies in these frameworks limit access to data that allow inference of this sensitive information. As the above examples show, today’s data rich world creates a new kind of problem: it is difficult if not impossible to guarantee that information does not

allow inference of sensitive topics. This makes information flow rules based on information topic unstable.

We address the problem of providing a workable definition of private data that takes into account emerging threats to privacy from large-scale data collection systems. We build on Contextual Integrity [4] and its claim that privacy is appropriate information flow, or flow according to socially or legally specified rules. As in other adaptations of Contextual Integrity (CI) to computer science (e.g., [5]), the parameterization of social norms in CI is translated into a logical specification. In this work, we depart from CI by considering rules that restrict information flow based on its origin and provenance, instead of on its type, topic, or subject. We call this concept of privacy as adherence to origin-based rules *Origin Privacy*.

Origin Privacy rules can be found in some existing data protection laws. This motivates the computational implementation of origin-based rules for the simple purpose of compliance engineering. We also formally model origin privacy to determine what security properties it guarantees relative to the concerns that motivate it.

2.1 Prior Work

Our work is related to standard notions of information flow in computer security in that we set out to track how information moves from the originating process to other processes. Traditional information flow in computer security is a descriptive property of a system; it models what it means for an information flow to exist from a system input to a system output (e.g., [6]). In contrast, origin-based privacy provides a normative theory: it explains which flows are appropriate, a decision considered as given by works on information flow.

Differential privacy, a probabilistic and quantitative notion of information flow [7], can be viewed as a method of “declassifying” information from an origin since it is implicitly an origin-based formalism. Thus, we use it as a technical mechanism for releasing data governed by an origin restriction.

Our approach is roughly dual to provenance (e.g., [8]): origin is to confidentiality as provenance is to integrity. Furthermore, while provenance-tracking systems may provide the building blocks of an origin enforcement mechanism, we also go further by having a normative theory of when and how to use it.

We can build risk-based approaches, such as NIST’s [9], on top of origin-based ones. In essence, if the risk of data misuse is too high for some data collected at an origin, people will withdraw from participating in that origin (e.g., by withholding substance-abuse information from their physicians). If the origin is deemed too important to society for people to withdraw from it, then the origin should be protected along the lines we propose to encourage participation.

Mosaic theory is a legal theory about how collecting many small pieces of data can be a privacy invasion even if each individual collection is not (e.g., [10]). It is similar to our position that one must be careful when releasing data collected in a particular context even if that data appears to be “non-sensitive”: such data may become enmeshed in a big data set that is able to reveal concerning facts about a person even if that data appears “non-sensitive” in isolation.

Privacy policies often restrict the purposes for which information may be collected (e.g., [11, 12]). Such restrictions are similar to our origin-based restrictions in that originating contexts are organized around a purpose. In essence, our approach can be viewed as a purpose restriction saying that data may only be used for the purpose of the context in which that data was collected.

Many policies (e.g., HIPAA, GLBA, FERPA, and Executive Order 13526 in the United States and the GDPR in the European Union) place restrictions on the collection, flow, and processing of personal information. When engineers build technical systems that collect and use personal data, they are under business and social pressure to translate prescriptive privacy policies, fitted to their case by lawyers, ethicists, and other policy-makers, into engineering requirements [13, 14, 15, 16]. The goal of engineering a privacy policy is to enable automated enforcement of some or all of the policy. This reduces the cost of protecting privacy. To automate enforcement of a privacy policy, that policy must first be translated into a machine-readable language with a precise syntax and semantics.

Prior research has explored the feasibility of translating classes of privacy clauses into formal logic for enforcement. Some attempt to formalize a wide range of policies, such as those expressible within the framework of Contextual Integrity [13, 17]. Others focus on particular kinds of clauses, such as those restricting information based on its purpose [18] or its use [19]. This article is concerned with clauses in privacy policies that restrict information based on its *origin*. We consider the origin of data to be the processes that created or transmitted that data to the system governed by the privacy policy, that is, the data’s provenance.

2.2 Structure

The rest of this report is structured as follows. We start by framing our view of what a privacy policy should do in Section 3.1. Section 3.2 lays out the basic building blocks information control for ensuring privacy. Section 3.3 quickly reviews Contextual Integrity and Origin Privacy. (More information on these privacy frameworks can be found in the work of Nissenbaum [4] on Contextual Integrity and our report for policy makers [20].) Section 3.4 uses causal models to explain these ideas in more depth. It shows how the informal specification of Section 3.3 can be mapped to a well established formal representation of causal models [21]. In addition to presenting the formal theory, we show, in Section 3.5, that causal modeling makes clear distinctions between two elements of information flow that are sometimes conflated: causal flow and nomic association, where “nomic” means law-like, or regular. These two aspects of information flow correspond to information origin and information topic. These models explain the ambiguity of rules prohibiting the flow of information.

Section 4.1 will show how existing policies motivate this work by defining restricted classes of information in terms of the processes that originated them. This policy analysis reveals that origin clauses are most often used to identify a broad class or type of information that is then subject to restrictions or exemptions. In addition, information topic (what the information refers to or is about) is also frequently used alongside information origin. We show that the distinction between information origin and information topic is subtle but important for the purposes of determining the conditions under which a formalized policy can be enforced.

In Section 4.2, we derive, from the policies analyzed, a general ontology of systems, processes, and messages. This ontology is intended to bridge between policies as intuitively articulated in natural language and the mathematical formalisms we will use in the rest of the paper. Using this ontology, we propose Origin Privacy as a framework for understanding privacy requirements and the knowledge necessary to enforce them by design.

Section 4.3 provides a syntax for our policy specification language. Sections 4.4 and 4.5 provide a semantics for the language. Section 4.6 covers enforcing policies with a type system, the

code for which we have already shared with DARPA [22].

In Section 4.7, we combine the ontology with causal modeling to develop the Embedded Causal System (ECS) model. This model represents a computational system embedded in a larger environment. This is motivated by the need to consider technical systems in their environments (possibly interacting with third parties) when assessing their privacy, fairness, and security properties [23]. We show demonstrate conditions under which an ECS model is secure according to the well-established formal security model of noninterference [6]. We also formalize semantic security for an ECS model and reproduce the result that it is, in general, impossible for a system designer to guarantee semantic security on a statistical database given auxiliary knowledge. It is well known that information can be revealing of other sensitive information given auxiliary knowledge. Our theorem reflects the conditions under which auxiliary knowledge is possible. The contributions from the model are due to explicit causal modeling of the generative process of data input into the system as well as the operations of the system itself.

In Section 4.8, we build on these results to develop security models for cases where information is restricted based on its origin. We find these models analogous to noninterference and semantic security, and demonstrate sufficient conditions under which an ECS has these properties.

Section 4.9 shows a case study of using Origin Privacy. We show that in a case of biometric sensing with Internet of Things devices, origin privacy specification can be used to enforce GDPR compliance.

Section 4.10 discusses how to select an origin-based privacy policy by looking at the incentives of the parties involved.

Section 5.1 looks at how to judge whether a policy is clearly specified. Section 5.2 considers how our limited knowledge about and ability to observe environments impact our ability to reason about privacy violations. Section 5.3 discusses how incentives impact privacy policy selection.

Section 6.0 addresses directions for future work. Sections 6.1 and 6.2 mention declassification and purpose-based restrictions as important areas for future research. Section 6.3 shows how differential privacy is a special case of origin privacy. Section 6.4 considers a direction for future research on policy authoring. It demonstrates how a game theoretic layer can be added to the ECS model to show how system security properties relate to the impact on system users.

This report refers to two technical appendices. Appendix A proves several supplemental theorems in information theory that are used in a proof of the relationship between causal modeling and differential privacy. Appendix B outlines the major findings of Koller and Milch [24] on Multi-Agent Influence Diagrams, and introduces a new concept: tactical independence. Also included is a note on how research supposedly building systems to protect privacy often lack justification that their approach is in fact providing privacy (Appendix C), which we have previously sent DARPA [25]. Lastly, we provide lists of our papers, both published (Appendix D) and unpublished (Appendix E).

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Theoretical frame

Privacy is a multifaceted concept [26]. In computer security, we typically concern ourselves with privacy as controls over the ways in which observers learn information about a data subject. While in general it is impossible to guarantee that a computer system releasing some useful data does

not lead to an adversary from learning a particular fact about a particular person [7, 27], computer systems can prevent certain actions impacting whether or not it happens [28, 7] and avoid obvious provisions of such information. A privacy policy can specify how to deploy such protections. To automate the enforcement of privacy policies in a general manner, those policies must be expressed to the enforcement mechanism to configure its operation.

The basic building blocks of privacy policies that limit the learning of information include access control, differential privacy, and Origin Privacy. Access control limits the release of certain data to particular entities. A system limiting its release of data cannot prevent, on its own, some entity from learning some fact since the entity might learn it from other sources. However, when access control denies the entity's request for particular data, it means that the system releasing that data will not be the cause of the entity learning that fact. If access control allows releases of other data, than that data could cause the entity to learn the fact. Furthermore, information can leak from a denial of a request. Access controls may depend upon the requested data, the requester, the requester's stated plans for the data, or other environmental factors such as the time of the request.

By limiting what outputs a system may produce, access control focuses on outputs with the goal of preventing the system's outputs from causing an adversary to learn some fact. Origin Privacy and differential privacy instead focuses on inputs. They limit the use of inputs by the system in its production of outputs with the goal of preventing an adversary from learning certain information about the inputs. Origin Privacy looks at where the input comes, its origin, to make these determinations. Differential privacy limits use in way that allows statistical inferences over a data set while not allowing any one data point to be revealed. In both cases, an entity providing an input can take comfort in knowing that its use will be limited.

Other possible building blocks may exist. Thus, for a policy language to long lived, it may need to be *extensible* to capture the understanding of privacy over time.

Given commonality of these basic building blocks to privacy and the sense that privacy is a single application of these building blocks, one may desire a general-purpose language for the specification of privacy policies. Such a language could then configure a general-purpose mechanism for enforcing access controls and differential privacy in a manner appropriate for a particular application. However, privacy is context specific [4] and such a language would have to capture every context, an impossibly broad task.

We instead search for a *parametric* family of languages for the specification of privacy policies where the parameter provides context-specific information. Whereas the extensibility mentioned before is additive and the language is usable without such an extension, the parametricity mentioned here represents a specialization of the language to a context, a necessary step before its use. While we believe our articulation of the need for a parametric family of languages is new, previous policy languages have included elements of parametricity or extensibility in their designs. For example, XACML allows an open-ended set of names for actions that could be taken on resources, which allows the set to depend upon the context of use.

Our approach to presenting such a language will be start with Legalease [16], an existing specification language for expressing privacy policies, and will modify it to operate with the aforementioned privacy building blocks. The first contribution in this write up is showing how Origin Privacy is comparable to the other building blocks of access control and differential privacy. The second is showing how Origin Privacy can be expressed in a modified form of Legalease.

3.2 Concepts of Information Control

A privacy policy consists of a series of rules. Each rule either allows or disallows some action. Since we are focusing on data privacy, these actions involve the use or sharing of data. Thus, we must start with a conception of data in motion.

Traditionally, computer security has defined information flow in terms of noninterference [6]. Recently, Tschantz et al. has shown that interference is equivalent to a notion of causation [29], which we'll use as our starting point. That is, we will consider some data X to *flow* to some other data Y if the value of the data X causes the value that the data Y takes on. Similarly, we say that an agent uses some data X if that that agent's behavior is caused by the value of the data X . In this way, the difference between data use and flow is minimal in our model.

More formally, let one variable Y be function another X along with other variables Z . We write this as $Y := F(X, Z)$ where we use the assignment notation $:=$ instead of the normal equality notation $=$ to remind us that this *structural equality* is directed in that the value of Y is defined in terms of X and Z , not the other way around. That is, changing the value of either X or Z may impact the value resulting value of Y , but changing the value of Y does not, in itself, impact the value of X or Z , as in the structural equations used by Pearl [30]. If changing the value of X does change the value of Y , we say that X *causes* Y . That is, if there exists values x_1 , x_2 , and z such that $F(x_1, z) \neq F(x_2, z)$, then X *causes* Y .

For example, let smoke, fire, and machine be booleans indicating whether a room has smoke, a fire, or an operating smoke machine, respectively, in it. $\text{smoke} := \text{fire} \vee \text{machine}$ is a reasonable assertion since having either a fire or a smoke machine in a room leads to smoke being in the room. (While reasonable, it is incomplete because other causes of smoke exist, but this incompleteness is true of nearly any model of reality.) Furthermore, changing whether or not there is a fire in a room causes smoke: adding fire to a room will typically make the room get smokey. And, the same holds for the smoke machine, justifying them assigning to smoke. On the other hand, $\text{fire} \vee \text{machine} := \text{smoke}$ is unreasonable since adding smoke does not cause fires to break out or machine to appear. (This is not just an artifact of aforementioned incompleteness of the model. We can think of no cause of smoke that we could add to the model that smoke also causes. The relationship between smoke and its numerous causes appears one directional.)

In general, this notion of causation is not transitive. For example, suppose that $Y := X$, $Z := -X$, and $W := Y + Z$. (That is, $Y := F_Y(X)$ where $F_Y(X) = X$, $Z := F_Z(X)$ where $F_Z(X) = -X$, and $W := F_W(X)$ where $F_W(Y, Z) = Y + Z$.) In this case, X causes Y and Y causes W , but X does not cause W . This can complicate reasoning in a compositional manner about causation. Thus, we will consider the transitive closure of causation as a compositional over-approximation of causation.

The first notion of privacy we have is that in some cases we would like to prohibit both a variable Y being caused by another variable X and X being shared with some entity. This can be expressed as an access control rule.

Requiring Y to not depend upon X at all is sometimes too strong of a requirement. We might desire that Y only depends upon X in a safe way. Of various notions of quantitative information flow exist that attempt to bound the degree to which Y depends upon X , the most interesting from a privacy perspective is differential privacy. It requires that Y depends upon a series X s, called $X_1, X_2, \dots, X_n$, with each X_i not contributing much to the value Y . To explain differential privacy in more detail, let us revisit the notion of causation and build upon it. Taking the negation of our definition of causation for such a sequence $X_1, X_2, \dots, X_n$, we get the requirement that for all values

$x_1, x'_1, x_2, x'_2, \dots, x_n, x'_n$, we have that

$$F(x_1, x_2, \dots, x_n) = F(x'_1, x'_2, \dots, x'_n). \quad (1)$$

We could relax this definition to be probabilistic: for all values $x_1, x'_1, x_2, x'_2, \dots, x_n, x'_n$ and y , we have that

$$\Pr[F(x_1, x_2, \dots, x_n) = y] = \Pr[F(x'_1, x'_2, \dots, x'_n) = y]. \quad (2)$$

Next, we can make the definition approximate: for all values $x_1, x'_1, x_2, x'_2, \dots, x_n, x'_n$ and y , we have that

$$\Pr[F(x_1, x_2, \dots, x_n) = y] \leq e^\varepsilon \Pr[F(x'_1, x'_2, \dots, x'_n) = y] \quad (3)$$

where ε is the degree of approximation. Finally, we can limit the approximate requirement to cases where only one pair of values x_i and x'_i differ: for all values $x_1, x'_1, x_2, x'_2, \dots, x_n, x'_n$ and y such that for all i in $\{1, 2, \dots, n\}$ except one value j , $x_i = x'_i$, we have that

$$\Pr[F(x_1, x_2, \dots, x_j, \dots, x_n) = y] \leq e^\varepsilon \Pr[F(x_1, x_2, \dots, x'_j, \dots, x_n) = y] \quad (4)$$

where ε is the degree of approximation. This requirement is equivalent to differential privacy for discrete probability spaces. Intuitively, it says that F will not use any one data point j more than a factor of e^ε . Tschantz et al. provide a more detailed explanation of the relationship between differential privacy and causation [31, 32].

More generally, we take the basic control applied by a privacy policy to be one that for some attributes X and $Y := F(X, Z)$, for all x and x' and z , $F(x, z) \approx F(x', z)$ where the approximate equality $\approx$ varies from definition to definition. For noninterference/causation, it is equality $=$. For differential privacy it is $\leq e^\varepsilon$, and the requirement holds for multiple attributes X (which are in Z when its not their turn).

On top of this basic building block, we add conditions. Conditions limit when such rules are applicable. For example, consider the privacy policy of a hospital. It might in general require complete noninterference from any patient's HIV status to any publicly available output, but it allows flows from a patient's HIV to a publicly available study under the restriction of differential privacy and under the condition that the patient opted into the study. We could view this policy as two disjoint rules:

1. Unless the patient opted in, total noninterference.
2. If the patient opted in, then differential privacy.

However, we find it easier to reason about a general rule and an exception:

1. Total noninterference except...
2. if the patient opted in, then differential privacy.

and will structure our policies as such.

While set of basic information flow restrictions is general and easy to capture with a handful of concepts (noninterference/causation and differential privacy might suffice), the set of conditions are open ended and dependent upon the context to which the policy applies. Thus, we now turn to contextual integrity.

3.3 Contextual Integrity and Origin Privacy

We conceive of privacy in a manner similar to that of Contextual Integrity [4]. Under Contextual Integrity, contextual informational (privacy) norms must list five independent factors:

- subject: the entity about whom the data is,
- sender: the entity transmitting the data,
- recipient: the entity receiving the data,
- transmission principle: pre- and post-conditions placing requirements on the transmission of the data (i.e., the aforementioned conditions), and
- information type: the topic of or sort of the data transmitted.

Origin Privacy is a modification of Contextual Integrity that replaces the subject and information type (topic) factors with references to the origin of the data in question. The motivation for Origin Privacy is that determining these factors is becoming increasingly difficult as machine learning and big data analytics makes increasingly unexpected inferences possible. Taken to the extreme, the notions of data subject and topic degenerates into every data point being about everyone and everything. The origin of data, however, only refers the past, the history of the production of the data in question. While the past might not be known to an entity determining the legitimacy of a proposed flow of data, it is a well defined concept that can be explored using traditional forensic methods. Furthermore, if adopted, provenance systems may be used to track history [E.g., 8]. Such features makes Origin Privacy policies more amendable to automated enforcement.

Under Origin Privacy the factors become

- origin,
- sender,
- recipient, and
- transmission principle.

Note that the concept of origin is more general than that of sender in that the origin stretches back in time rather than just referring to the immediate sender of the data. Also note that Origin Privacy does not, on its own, ensure that a given privacy policy can be enforced using automated means. Origin Privacy only addresses the difficulties posed by subjects and topics; additional difficulties may exist in enforcing the transmission principles of a privacy norm.

As an example of applying the ideas of Origin Privacy, consider part of Google’s privacy policy from 2013, which can be represented as in Figure 1 in a syntax that we will make precise later. The “DataType” field of “PII” is a problematic subject/topic. Thus, under Origin Privacy, we replace it with by an origin-based field to get the policy in Figure 2. Note that translation, motivated by Origin Privacy, does not address the difficulty of enforcing purpose-based conditions [11, 18].

```
(ALLOW EXCEPT
  (DENY (DataType PII) (UseForPurpose Sharing) EXCEPT
    (ALLOW (DataType PII:OptIn))
    (ALLOW (AccessByRole Affiliates))
    (ALLOW (UseForPurpose Legal))))
```

Figure 1: Example policy making use of data types

```
(ALLOW EXCEPT
  (DENY (Origin users) (UseForPurpose Sharing) EXCEPT
    (ALLOW (Origin users:OptIn))
    (ALLOW (AccessByRole Affiliates))
    (ALLOW (UseForPurpose Legal))))
```

Figure 2: A policy example making use of origin in place of data types

3.4 Information flow and causal models

We have motivated Origin Privacy as a concept of privacy that is useful when considering how to design information processing systems to be compliant with laws and other rules regarding the flow of personal information. The ontology in Section 4.2 is motivated by policies that specifically mention systems and restrict information flow based on its origin.

In this section we will introduce philosophical and formal concepts with which we will make our definitions and claims about origin privacy precise. Philosophically, privacy depends on appropriate information flow [4], where information is defined as that which allows somebody to learn about something else based on its regular associations with it. We find a formalization of this idea in Bayesian networks, a common formalism in statistics which represents the relationships between random variables with a directed acyclic graph. Bayesian networks have two attractive properties which we will explain. First, it is easy to derive some independence relations between variables from the graph structure of a Bayesian network. Second, this formalism supports an intervention operation that gives it robust causal semantics. All of these conceptual tools will be used in proofs later in this paper. We will close this section by showing how this formalism rigorously clarifies an ambiguity in the term ‘information flow’, which refers to both causal flow and nomic associations between variables. We adopt the term *situated information flow* for this sense of information flow in causal context.

3.4.1 Philosophy: contextual integrity and information flow

Origin Privacy is intended to be broadly consistent with the contextual integrity [4] philosophy of privacy in so far as it defines privacy as *appropriate information flow*. Specifically, contextual integrity maintains that privacy expectations can be characterized by norms about how information about persons flows in particular social contexts, or spheres.

In this article, we restrict our analysis of Origin Privacy to cases where expectations of privacy have been articulated as *policies* in natural language and endowed with a social system of enforcement. We consider laws and contracts as kinds of policies. Policies may or may not express social

norms as per contextual integrity; addressing the conditions under which policies reflect social norms is beyond the scope of this article. However, we maintain that some policies are specifically *privacy* policies because they, like privacy norms, prescribe personal information flows.

As a way of bridging from contextual integrity through privacy policies to the specification of privacy-preserving mechanisms, we address *information flows* in general. Despite its wide use, the phrase “information flow” is rarely given a precise definition. However, philosophical and formal work on information flows have provided general insights that can bridge between social, legal, and technical theories of privacy. We will build on these insights to make arguments about origin privacy.

There is a long history of literature on information flow in computer security and privacy research [33, 34, 35, 29, 36]. These take their inspiration from Shannon’s classic formulation of information theory [37]. Dretske’s philosophical formulation of information flow [38] also draws on Shannon’s information theory. In this work, we are explicitly bridging between philosophy and engineering principles, finding common ground between.

According to Dretske’s theory, a message carries information about some phenomenon if a suitably equipped observer could learn about the phenomenon from the message. In other words, a message carries information about anything that can be learned from it. For an observer to learn from it, the message must have a *nomic* connection with its subject, where here “nomic” means “law-like” or “regular” [39]. Messages, in this understanding, get their meaning from the processes that generate them, because these processes connect the content of messages reliably to other events. There is a logical connection between the definition of information flow and the structure of the regular dependence between events. The formal theory of the causal dependence between events has been worked out in the literature on causal graphical models [21].

3.4.2 Causal probabilistic graphical models

There is the a well known formalism for representing the causal relationship between uncertain events: the *Bayesian network*, or probabilistic graphical model, framework [21]. As we will see, this form of modeling can be used to represent processes within a system, as well as in its environment. Before showing the relationship between Bayesian networks and Origin Privacy, we will present them and a few of their formal properties, drawing heavily on Koller and Milch [24] for our choice of formal notation and wording.

A Bayesian network represents the joint probability distribution of a set of random variables with a graph. Consider variables $X_1, \dots, X_n$ where each X_i takes on values in some set $dom(X_i)$. We use $\mathcal{X}$ to refer to the set $X_1, \dots, X_n$ and $dom(\mathcal{X})$ to refer to their joint domain.

A Bayesian network (BN) represents the distribution using a graph whose nodes represent the random variables and whose edges represent direct influence of one variable on another.

Definition 1 (Bayesian network). A Bayesian network over variables $\mathcal{X} = X_1, \dots, X_n$ is a pair (G, Pr) . G is a directed acyclic graph with n nodes, each labeled for one of the variables in $\mathcal{X}$. We use $Pa(X)$ to denote the parents of X in the graph. Pr is a mapping of each node X to a conditional probability distribution (CPD), $Pr(X|Pa(X))$.

Example 2. (Figure 4.1) Alice will be on time for work W if she sets her alarm A early enough and traffic T allows. Bad traffic can be caused by construction C or an accident on the road D .

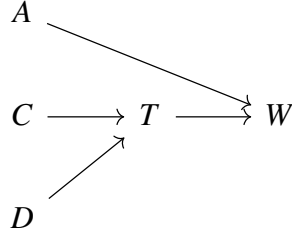


Figure 3: Alice's commute to work

Given the discussion of information flows above, we can see why this is relevant to privacy. The conditional dependence functions between random variables are the *nommic relations* between events and messages. If two variables are conditionally dependent on each other, and this conditional dependence is known to the observer of one of the variables, then the observer can infer something (have knowledge of) the other variables. Hence, by our definitions, the variables carry information about each other. If privacy is appropriate information flow, then the privacy of a system will depend on the causal relationships between its components and the environment.

A directed edge between one variable and another indicates a possible conditional dependence between them. Strictly speaking, it does not necessitate that there is a conditional dependence between them, it only necessitates that there is a conditional probability distribution function defined between them. But it does guarantee that at least one such conditional probability distribution does exist, and under reasonable conditions *most* possible functions (in a measure-theoretic sense) will exhibit the conditional independence [40]. As functions ensuring independence are quite rare in the space of all possible conditional probability functions, this quirk in the notation has not prevented this formalism from being useful in identifying independence in practice.

D-separatedness A useful property of probabilistic graphical models is that some aspects of the joint probability distribution of all variables represented in the graph can be read easily from the graph's structure. Of particular interest in the analysis of the joint probability distribution is when and under what conditions two random variables are independent.

Definition 3 (Path). A *path* between two nodes X_1 and X_2 in a graph to be a sequence of nodes starting with X_1 and ending with X_2 such that successive nodes are connected by an edge (traversing in either direction).

Definition 4 (Head-to-tail, tail-to-tail, head-to-head). For any three nodes (A, B, C) in succession on a path, they may be *head-to-tail* ($A \rightarrow B \rightarrow C$ or $A \leftarrow B \leftarrow C$), *tail-to-tail* ($A \leftarrow B \rightarrow C$), or *head-to-head* ($A \rightarrow B \leftarrow C$).

We will find it useful to refer to a special kind of paths, *direct paths*.

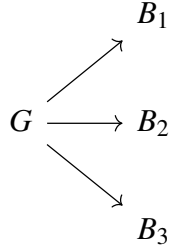
Definition 5 (Direct path). A *direct path* from X_1 to X_2 is a path starting with X_1 and ending with X_2 such that all triples are head-to-tail.

Definition 6 (Ancestors and descendants). If there is a direct path from X_1 to X_2 , then X_1 is an ancestor of X_2 and X_2 is a descendant of X_1 .

Let $\text{descendants}(X)$ be the set of descendants of X .

There are two ways in which a variable A can be conditionally dependent on another variable B without one of them being a descendant of the other. The variables may share an unobserved common cause or they may share an observed common effect.

Example 7. One building in a neighborhood loses power, B_1 . One can guess that other buildings B_i around nearby lost power, because power in each building is dependent on the electric grid G . All the buildings may be affected by the common cause of a grid failure.



Example 8. (Figure 4.1) Suppose we observe that Alice is late for work W , as per our earlier example. This could be due to many reasons, including traffic T and missing her alarm A . Traffic may be due to construction C or an accident D . The probability of any particular cause is conditionally dependent on the others, because if any one cause is ruled out, the others are more likely.

The existence of a path between two nodes is necessary for their probabilistic dependence on each other. It is not sufficient, particularly when considering their dependence *conditional on other variables*. For this reason, paths in a Bayesian network can be blocked or unblocked based on a set of variables that is otherwise known or observed, the *conditioning set*.

Definition 9 (Blocked path). A path is considered to be *blocked* if either:

- it includes a node that is in the conditioning set C where the arrows point to it do not meet head-to-head, or
- it includes a node where arrows do meet head to head, but neither this node nor any of its descendants is in the conditioning set

Definition 10 (D-separation). If every path from X_1 to X_2 given conditioning set C is blocked, then X_1 and X_2 are d-separated.

Theorem 11. If X_1 and X_2 are d-separated conditioned on set C , then $X_1 \perp\!\!\!\perp X_2 | C$.

Proof. Proof is discussed in Koller and Milch [24]. □

The converse (that independence implies d-separatedness) is not true in general because specific conditional distribution functions can imply independence. Similarly, it is not generally true that the absence of d-separatedness implies conditional dependence. However, it has been shown that conditional distribution functions implying conditional independence are rare in a measure-theoretic sense [41, 40, 24].

Intervention We have used the terms *Bayesian network* and *causal model* interchangeably. This is because Bayesian networks support a causal interpretation through one additional construct, *intervention* [42]. An intervention on a Bayesian network sets the values of one or more of its values. Unlike an observation of a variable, an intervention effectively creates a new graphical model that cuts off the influence of a set variable on its parents and vice versa. Descendants of the set variable are affected by the intervention according to the probability distribution of the original model.

Definition 12 (Intervention). An *atomic intervention* setting variable X_i to x'_i on a Bayesian network $\mathcal{W}$ creates a new network $\mathcal{W}'$ with post-intervention probability distribution $Pr_{x'_i}$

$$Pr_{x'_i}(X_1, X_2, \dots, X_n) = \begin{cases} \frac{Pr(X_1, X_2, \dots, X_n)}{Pr(X_i = x'_i | Pa(X_i))} & \text{if } X_i = x'_i \\ 0, & \text{otherwise} \end{cases}$$

Theories of causation based on manipulation and intervention have been influential in philosophy [43] and have been shown to be effective theories in psychology of causation [44] including the role of causation in moral reasoning [45], suggesting interventionist causation as a potential bridge between computer science and ethical domains such as privacy and fairness. Cowgill and Tucker [46] discuss the evaluation of algorithmic impact using counterfactuals, which draws on a different but compatible theory of causality [47].

3.5 Ambiguity of information flow

We have drawn a connection between information flow in the philosophical sense relevant to Contextual Integrity and Bayesian networks. A Bayes network is a way of representing the nomic dependencies between phenomena. They are “nomic” because they describe probability distributions that generalize over particular instances of a system’s functioning. These nomic relations are factored out as an explicit structure of causal relationships.

This reveals an ambiguity in the very concept of *information flow*, illustrated in the following example.

Example 13. Alice, a teacher tells every student privately their test score’s rank R (first in class, second in class, etc.) after every test, with class participation used as a tie-breaker. Alice sends a message B to Bob with the information that he has the second highest rank in the class. Alice also sends a message E to Eve that she has the highest rank in the class. From her message and knowledge of the test environment, Eve learns from her message that Bob was told that he was, at best, second in class. Did information about Bob flow to Eve?

A formal representation of this example makes the root of the ambiguity clear. Consider a three node Bayesian network where R is the test results, B is the message sent to Bob, and E is the message sent to Eve (fig. 4).

There is causal flow along the edges from R to B and from R to E . But an observer of a single variable aware of the system’s laws (nomic connections, graphical structure) can learn nomic associations of a message that inform about variables that are not in the message’s causal history. Despite E neither causing nor being caused by B , E reveals information about B .

The phrase “information flow” is ambiguous because the word “information” is ambiguous [48]: it can refer to both a message and the contents of a message. We do not favor either sense.

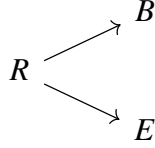


Figure 4: Test score ranks (R) distributed to Bob (B) and Eve (E)

Rather, we propose that to resolve this ambiguity, one has to recognize how the systematic creation and transfer of messages—represented in general by a graph of causal flows—gives each message its meaningful contents. In our formalism, a situated information flow is a causal flow that, by virtue of its place in a larger causal structure, has nomic associations.

Our analysis of privacy policies shows how they variously restrict the flow of information based on its contents as well as its causal history or origin. This is consistent with our analysis of “information flow” as referring to one causal flow within a larger system of causes that give it contents. This scientific formulation of information flows is not yet native to the language of the law. That the law refers variously to aspects of information flow based on contents and causal history reflects how both are essential to the meaning of the term.

In the following sections we will precisely model a system in its environment in order to disambiguate the different aspects of information flow and understand the conditions under which a system can be free of information leaks. We will measure the strength of nomic associations using a well-understood measure, *mutual information*. Mutual information captures how much about one random variable can be learned from observing another.

Definition 14 (Mutual information). The mutual information of two discrete random variables X and Y is

$$I(X, Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

In particular, $I(X, Y) = 0$ if and only if $X \perp Y$.

Mutual information is a technical term with a specific mathematical meaning. It is no etymological accident that it forms part of the analytic definition of “information flow” that we have developed in this section. In the Appendix A, we derive several theorems relating to the mutual information of variables organized in a Bayesian Network. We will use these theorems in proofs about system security in the following sections.

4.0 RESULTS AND DISCUSSIONS

4.1 Policy motivations

To motivate a consideration of Origin Privacy as a flavor of privacy specification, we look to existing policies that include rules that restrict information flow based on its origin. We build on prior work in logical specification of privacy laws as inspiration for this approach [5, 49].

4.1.1 Policy example: HIPAA

Consider as an example, the following information flow rule, a vast simplification of the types of rules found in HIPAA, intended to illustrate the difficulties of topic- and subject-based rules:

Health information about a patient will not be disclosed by the provider except with that patient's permission or with a court order.

This rule references the topic “health information” and the data subject, called by his role of “patient”. An automated enforcement mechanism determining whether a flow of information is allowed under this rule would have to know whether the data is “health information” and, if so, about which patient(s). The mechanism could attempt to determine whether the data is medical information by examining the data and any other available information, such as meta-data about the data.

However, determining data's topic and subject are difficult. For example, since diet predicts a large number of health issues, any comprehensive conception of topic would have to include a patient's diet as health information despite such information appearing “non-sensitive” at first glance. Further, suppose that big data analytics finds an unexpected strong correlation between one's address, or TV viewing habits, and a health condition. Then, a comprehensive treatment of medical information would have to include this data despite it being difficult to predict that they would have any bearing on one's health status.

Being unable to predict such discoveries, to ensure that the rule is always obeyed under a comprehensive conception of topic, an enforcement mechanism would have to err on the side of caution and treat nearly any data as health information. Similarly, any comprehensive treatment of the data's subject(s) would have to err on the side of considering the data to be on everyone.

Given that a comprehensive treatment of topic and subject leads to enforcement mechanisms that treat topic and subject restrictions degenerately, one might consider alternative conceptualizations of topic and subject. For example, one could state that the topic of data is determined by its syntactic form, for example, whether it includes standard medical terminology. However, such a syntactic approach could be easily gamed by those who wish to obey the letter of the privacy rule but not its spirit, for example, by rewording the data. Alternatively, one could try to intuit the topic of data as what the data is “directly” or “primarily” on. However, even if this conception is sound, we know of no way to automate such a determination. Lastly, one could argue that the metadata might include the data's topic and subject. However, this is begging the question: any process that supposedly creates the metadata would run into the same problems as the enforcement mechanism.

Thus, we take a different approach and replace references to the data's topic or subject with references to how the data was created, that is, its origin. As a first approximation, under origin privacy, we will rewrite the above rule as the following:

Data originating from a patient's examination at a healthcare provider will not be disclosed except with that patient's permission or a court order.

This rule focuses on the data's source, or origin, as an interaction between a patient and healthcare provider, instead of its topic. Furthermore, rather than focusing on data about that patient viewed as a subject, the rule focuses on data produced by that patient's interaction, regardless of whether it is about that patient.

For such a system to be workable, the above rule will need refinement. In particular, not all medical data collected by healthcare providers are governed by the same norms. For example, a patient's religious affiliation may, under HIPAA, be shared with clergy to ensure pastoral care. A patient's religious affiliation, as commonly understood, is information about data subject on the topic of religion. However, under origin privacy, we will instead sub-divide the origin into sub-origins and refine the rule as follows:

Data originating from a patient's examination at a healthcare provider will not be shared outside that provider's office except with that patient's permission, a court order. As an exception, if the data was created in response to a query for the patient's religious affiliation, it may be shared with clergy without patient permission..

The refined rule avoids invoking a topic by instead referring to how the data was collected, whether it was offered by the patient in response to a solicitation for their religious affiliation.

Instead of the data's topic and subject, a mechanism enforcing this rule for a piece of data needs access to metadata recording whose examination produced the data and process by which it was collected and entered. Since such provenance metadata can easily and automatically be recorded by electronic medical record systems, giving such metadata to the mechanism is straightforward unlike determining the topic or subject of data.

To see how well this approach can HIPAA, we examined HIPAA in detail. We chose HIPAA as a starting point since it is a well studied law. Our goal is to first ensure that Origin Privacy behaves as expected in a well understood case before applying it to laws with uncertain outcomes.

Our examination looked for two factors. First, we looked for rules in HIPAA that were already expressed in terms of the origin of data. Second, for those rules that were expressed in terms of topics or subjects, we attempted to find techniques to convert the rule into one expressed in terms of the data's origin instead. We present one such example below. We also present an example in which the reference to an topic does not pose problems requiring its conversion.

A straightforward example of HIPAA defining a class of information in terms of the process that creates it is psychotherapy notes. HIPAA defines them as follows:¹

Psychotherapy notes means notes recorded (in any medium) by a health care provider who is a mental health professional documenting or analyzing the contents of conversation during a private counseling session or a group, joint, or family counseling session and that are separated from the rest of the individual's medical record. Psychotherapy notes excludes medication prescription and monitoring, counseling session start and stop times, the modalities and frequencies of treatment furnished, results of clinical tests, and any summary of the following items: Diagnosis, functional status, the treatment plan, symptoms, prognosis, and progress to date.

In this definition, there is a reference to a process involving "documenting or analyzing [...] a [...] counseling session". Any information with provenance beginning with its creation by a health care provider who is a mental health professional documenting a conversation during a counseling session separately from an individual's medical record, barring some exceptions, are psychotherapy notes.

¹45 CFR §164.501.

Many rules in HIPAA refer to match between the identifier of the data's subject and the identifier of the data's recipient, such as in cases where a patient may access her own data. In these cases, we can convert these rules into ones that refer to the patient that provided the data instead of the data's subject. Such a rule makes more sense given how hospitals actually treat medical information under HIPAA. For example, when a patient tells her physician about a her family's medical history, it goes into that patient's record, accessible to her, instead of the records of her family members [50]:

When a covered health care provider, in the course of treating an individual, collects or otherwise obtains an individual's family medical history, this information becomes part of the individual's medical record and is treated as 'protected health information' about the individual. Thus, the individual (and not the family members included in the medical history) may exercise the rights under the HIPAA Privacy Rule to this information in the same fashion as any other information in the medical record, including the right of access, amendment, and the ability to authorize disclosure to others.

This shift would also make it easier to operationalize the enforcement of the privacy policy. To do so, an enforcement mechanism would just need to ensure that the electronic medical record system consistently identify patients, records which data was provided by which patients, and checks for a match of patient to recipient identifiers.

We can generalize the above policy transformation from just patients (which wouldn't make sense for some contexts) to the person in the protected role who provided the data. We limit the protected origin to a protected role (e.g., patient, student, bank customer) to apply the privacy protections to the intended people rather than, for example, employees of the system governed by the policy. This limitation prevents the policy being used to withhold, for example, a physician's notes created while treating a patient on the grounds that the physician provided the notes.

In a second example, consider that HIPAA, based on 45 C.F.R. §164.510(a), allows a covered entity to "disclose the individual's [general] condition and location within the facility to anyone asking for the individual by name" [51]. The patient's condition and location are data topics referred to by HIPAA.

However, our concerns about the semantics of topics do not apply to this rule since it is an exception allowing the sharing of data to a more general prohibition against sharing. Mechanisms enforcing such exceptions can safely use a simple syntactic check for whether the provided data fits the exception's topic in a syntactic sense. For example, it could check that the condition provided is just one of the following words: "dead", "critical", "stable", "recovering", or "discharged". Such a check could produce false negatives in the sense that other data, such as "deceased", can also convey a patient's general medical condition. However, a healthcare provider wishing to provide this information can reword its response to fit this syntactic requirement (and also generalize the above list). Such a simple check could also produce false positives. For example, the word "stable" might refer not to the patient's general condition, but rather to detailed information out his blood sugar. However, providing an allowed answer to the wrong question would constitute a more clear-cut violation of either the privacy policy or the specification of a correctly operating system, making it relatively easier to catch in an audit. Thus, since the rule is an exception allowing the sharing of data, there is no privacy risk of the data being reworded to avoid the rule's prohibition on sharing.

Nevertheless, information can be unexpectedly leaked under this exception. For example, a room number could convey a health issue in a hospital that treats different conditions in different rooms. This is a threat to the ability of a hospital to abide by both the traditional promise to not share medical information and to an origin-based one not to share information learned during an examination (e.g., since information collected during the examination might have determined the assigned room). This threat means that this exception has privacy consequences beyond a patient's condition or room number, and that the policy authors must carefully weigh these consequences with the benefits of sharing this information. Our work will highlight these consequences.

4.1.2 Policy example: GLBA

Some laws include references to information origin in the definition of what information is protected. We find this in particular in the Privacy Rule of the Gramm–Leach–Bliley Act, which applies to “nonpublic personal information” (NPI).² This class of information is defined as personally identifiable financial information that is

1. provided by a consumer to a financial institution;
2. resulting from any transaction with the consumer or any service performed for the consumer;
or
3. otherwise obtained by the financial institution.

Reading progressively through each of these criteria, the concept of *origin* helps us understand the differences between them and how that affect their enforceability. Criterion 1 explicitly refers to the channel of transmission and does not refer to any specific meaning or category of the information transmitted (though examples are provided in the law, these are constrained only by what is normally transmitted in the process of procuring a financial service). It sets clear guidelines as to the process of creation and transmission. Criterion 2 refers to broad classes of ways in which the information is transmitted to the governed entity. Criterion 3 is written as if to cover all other cases where a financial institution could collect individualized information, regardless of the process of transmission. It is agnostic to the process of transmission. It raises the question of whether information can be personal financial information without having the specific provenance of a transaction or service with a financial institution. For example, information about a person's income may be personal financial information no matter how it is discovered.

4.1.3 Policy example: PCI DSS

Though not a law, we consider the Payment Card Industry Data Security Standard (PCI DSS) to be a security regulation [52]. Established as a proprietary information security standard by the Payment Card Industry Security Standards Council, it applies to organizations that use major branded credit cards such as Visa, Mastercard, and American Express. Though referenced in the laws of some U.S. states, it is mandated mainly by Visa and Mastercard themselves through their dealings with merchants.³

²GLBA, 15 U.S. Code §6809

³Minnesota Session Laws - CHAPTER 108–H.F.No. 1758. Nevada Revised Statutes, Chap. 603A §215. Wash. Rev. Code §19.255.020 (2011). See [53]

We note two aspects of the PCI DSS that make it an effective regulatory standard. First, the PCI DSS governs only types of data the enforcing industry is responsible for generating: cardholder data and sensitive authentication data [54]. This data is generally not meaningful outside of the operations that the payment card industry enables, because the potential uses of the data are a function of the system that created the data in the first place. This allows the payment card industry to straightforwardly enforce contractual obligations on those that use the data. This is in contrast with legal regimes that regulate the flow of more generally meaningful information between persons and organizations.

A second feature of PCI DSS is that it is explicit about the application of the standard to networks of technology and business processes, which it calls the *cardholder data environment* (CDE) [54]:

The PCI DSS security requirements apply to all system components included in or connected to the cardholder data environment. The cardholder data environment (CDE) is comprised of people, processes and technologies that store, process, or transmit cardholder data or sensitive authentication data.

which PCI DSS further defines as [54]:

The first step of a PCI DSS assessment is to accurately determine the scope of the review. At least annually and prior to the annual assessment, the assessed entity should confirm the accuracy of their PCI DSS scope by identifying all locations and flows of cardholder data, and identify all systems that are connected to or, if compromised, could impact the CDE (for example, authentication servers) to ensure they are included in the PCI DSS scope. All types of systems and locations should be considered as part of the scoping process, including backup/recovery sites and fail-over systems.

This is a clear example of how a privacy policy can be specific about the technical system to which it applies.

4.1.4 Other policies

The examples above have been chosen for their representativeness of the concepts developed in this paper. Other information protection policies do not define protected information solely in terms of its origin but rather depend in whole or in part on a definition of information topic. For example, the Family Educational Rights and Privacy Act (FERPA) defines “education records” as

those records, files, documents, and other materials which—

- (i) contain information directly related to a student; and
- (ii) are maintained by an educational agency or institution or by a person acting for such agency or institution.

The Children’s Online Privacy Protection Rule (COPPA) includes a broad definition of “personal informal” as any individually identifiable information collected on-line, but includes special restrictions on personal information collected from a child, which could be read as a restriction based on origin.

The United States Executive Order 13526 of 2009 gives certain government officials authority to classify documents they determine pose a risk to national security. In some cases, the information may be classified as soon as it is produced. In all cases, the information's classification prevents unauthorized access. Derivative information carries the classification of the original information. Information may be declassified when the conditions for classification no longer hold. Information restrictions on information therefore depend partly on its procedural history, but also on predictions made about the effects of disclosure.

Our analysis of Origin Privacy explores the limits of privacy by design and what policies can be automatically enforced on a system bound by laws of statistical causation. We will demonstrate the ambiguity of the term “information” and how this renders the application of many policies that restrict information based on information topic indeterminate.

4.2 Origin Privacy

We will now provide an informal definition of Origin Privacy. First, we will introduce an ontology appropriate to the design of technical systems and inspired by the policy examples above. Three concepts are introduced in this section. *Systems* are assemblages of people and technologies through which information flows and is transformed. *Processes* are regular events, implemented as a person or technology's behavior, which act on information. Processes pass information to other processes as data. The *origin* of data is the history of processes that led to its creation. Origin privacy includes any privacy restrictions made on information flow conditioned on that information's origin.

4.2.1 Systems

Regulations mark out spaces or domains in which information may flow more freely than others, or where restrictions specifically apply. For example, HIPAA applies to “covered entities”, including health care providers.⁴ These spaces may be defined by legal jurisdiction, or they may be defined through networks of contractual relations. More often than not, the regulated space is not bounded geographically but rather by relationships between personnel, institutions, and technical systems, all of which are participating in information flows and processing.

The Payment Card Industry Data Security Standard (PCI DSS) is explicit about this in its definition of the covered system in terms of the “people, processes, and technologies that store, process, or transmit” data (Section 4.1.3 provides further details) [54]. We generalize this concept and refer to assemblages of people, processes, and technologies such as these *information processing systems*, or just *systems*.

There will inevitably be people, processes, and technologies that interact with a governed system without being included within it. We refer to these external factors as the *environment* of the system. Systems have inputs and outputs, which are events that pass data from and to the environment. We will find it useful to model the world of a system within its environment as a larger, superordinate system. This causally embedded system framework is formalized in Section 4.7.1.

⁴“Covered entity means: (1) A health plan. (2) A health care clearinghouse. (3) A health care provider who transmits any health information in electronic form in connection with a transaction covered by this subchapter.” 45 CFR §160.103

The use of the term “system” here is abstract and intended to provide a precise analytic frame. How it gets applied to an empirical case can vary. For example, in healthcare, we might consider a single hospital with its procedures as a system, or a network of covered entities collectively as a system. A complete discussion of systems and how they may be nested or interconnected is beyond the scope of this paper.

4.2.2 Processes

For the purpose of these definitions, we will assume that technical systems and their environments consist of many different components implementing information processes. People in their professional roles are, for the purposes of this framework, included as another way of implementing processes.

The outputs of a process may be the input to another. These messages between processes are *data*. When a process *A* sends data to another process *B* under one or more conditions, we say that there is a channel from *A* and *B*, or, equivalently, that (the state of) *A* directly causes (the state of) *B*.

The structure of processes and their dependence on each other implies a causal graph [21] with directed edges representing the channels between processes. We will discuss the implications of causal modeling of systems more thoroughly in Section 3.4.

Systems are composed of contiguous processes, meaning that for any two processes in the system there will be an (undirected) path between the two through channels that includes only other processes in the system.

While causal modeling has been used in a security context (e.g., Feng et al. [55]), formal security research more often relies on static analysis of programs (e.g., McLean [33]). We choose a causal models in this work because these models can represent both programs and their subprograms, as well as non-technical environmental processes that generate data.⁵ An example of such a process is a medical examination conducted via an interpersonal interaction between patient and doctor.

4.2.3 Origin and provenance

The data resulting from a process depends causally on a history of prior processes. Sometimes data that is an input to a system is generated by a process that is not immediately part of the system. Data can flow through a series of relays before it reaches the system on which a privacy policy is being enforced. The entire history of the information as it flows from its creation to the system input is the information’s *provenance*. The governed system may or may not have access to assured metadata (such as cryptographic certificates) about the provenance of its data.

We consider the origin of data to be the processes that have caused it, either directly or indirectly. For the purposes of enforcement of a policy, these processes may be either in the governed system or outside of it, in an originating system or the system’s environment.

⁵While it may be the case that programs are as expressive as causal models, this discussion is beyond the scope of this paper.

4.2.4 Origin privacy

Given the above ontology, we can now provide a definition of origin privacy. Origin privacy includes any and only those information flow restrictions implemented in a system that are conditioned on the provenance of system inputs.

4.3 Syntax

Our work will build off of prior research on enforcing policies in the privacy domain. Sen et al. consider auditing the use of information in real systems where policies must be expressed in an easy to understand language and data sources are unlabeled [16]. They provide the *Legalease* policy language, designed for use by lawyers. We use this language as a starting point.

Bearing in mind that privacy is context dependent, we should not expect a single language to aptly capture every privacy policy. Thus, our specification language consists of two parts: a description of the context and the policy itself. The description instantiates a parametric family of policy languages into a domain-specific language for the context at hand by providing context-specific attributes for use in the policy. Although one could view the description as part of a policy, by describing the world, instead of constraining how it should operate, the description should be analyzed using empirical rather ethical considerations, unlike the policy proper.

For purposes of providing a system of automated enforcement, we will view a description as a mapping from attributes referred in the policy to tests that can be ran to determine whether they hold. The correctness of the description depends upon whether the tests capture the meaning of the attributes and whether attributes capture the nature of the context. The syntax for descriptions is

Description	$D ::=$	$(\text{DESCRIPTION } M_1 \cdots M_n)$
Attribute Definition	$M ::=$	$(N F) \mid (N F S_1 \cdots S_n)$
Attribute Name	$N ::=$	text
Function	$F ::=$	function that determines the value of the named attribute
Subtyping	$S ::=$	$(V_1 V_2)$

We allow ? to be used in the place of a concrete function F to denote the absence of a function for determining the attribute's value. Such attributes must be check manually. Prior by Garg et al., [56] which study policy auditing over incomplete logs, can be applied here to minimize the number of required manual checks. A subtyping clause $(V_1 V_2)$ shows that a value V_2 is a subtype of V_1 for a attribute.

The syntax for the policy proper is

Policy Clause	$P ::=$	D
Deny Clause	$D ::=$	$(\text{DENY } T \text{ EXCEPT } A_1 \cdots A_m)$ $\mid (\text{DENY } T)$
Allow Clause	$A ::=$	$(\text{ALLOW } T \text{ EXCEPT } D_1 \cdots D_m)$ $\mid (\text{ALLOW } T)$
Target	$T ::=$	$C_1 \cdots C_n$
Condition	$C ::=$	$(N V)$
Value	$V ::=$	attribute-value

As syntactic sugar, if the value V is a tuple, we allow each component to be listed after the attribute name without enclosing parentheses. If no values are provided, we take it to be implicitly the value of true.

The conditions can either refer to attributes defined by a description associated with the policy or to a primitive term: `noninterference` or `differential_privacy`. The semantics of these primitives are common across contexts and built into the semantics of the language, making their explicit description unneeded.

As an example, consider the description and policy in Figure 5.

```
(DESCRIPTION
  (Origin lambda data: sources(get_metadata(data))
           (users users:OptIn))
  (UseForPurpose ?)
  (AccessByRole current_roles))

(ALLOW EXCEPT
  (DENY (Origin users) (UseForPurpose Sharing) EXCEPT
    (ALLOW (Origin users:OptIn))
    (ALLOW (AccessByRole Affiliates))
    (ALLOW (UseForPurpose Legal))))
```

Figure 5: Example attribute and policy specification

The description states how to determine the origin of data from its metadata. This is feature of the particular system governed by this policy. Similarly, it states how to determine the role of the person attempting to access the data. It does not explain how determine the purpose of the proposed data use; this must be check by hand. The description also shows that `users:OptIn` is a subtype of `users`.

The policy allows any access except those accessing data originating with users for the purpose of sharing. Those access are only allowed when user opts in, the access is by an affiliate, or the purpose is legal. This shows that adjacent conditions `(Origin users) (UseForPurpose Sharing)` are implicitly conjoined while the adjacent allow rules are disjointed.

4.4 General Semantics

We explain the semantics of a policy in terms of an entity attempting to gain access to a resource. Such an entity makes a *request* for access. The request captures the entity’s identity, the resource requested, and any other information relevant to the decision. A *policy evaluation engine* compares the request to the policy to determine whether to grant access to the entity as proposed in the request. Thus, the semantic meaning of a policy is, roughly speaking, a function from requests to decisions to either allow or deny the request.

4.4.1 Requests

In the discussion of policy semantics below, we model requests as syntactic objects created at the time the entity made the request, which is before its comparison to the policy. However, for ef-

efficiency, the policy evaluation engine doing the comparison of the request to the policy can start with an incomplete request and construct the rest of it on the fly based upon the attributes in the policy whose values must be known to complete the comparison of the request to the policy. In cases where the partial request is already sufficient to determine that it will surely be allowed or surely denied regardless of the values of the missing attributes, the policy evaluation engine can skip determining those attributes' values. When it comes to attributes whose definition is $?$, indicating the need to for a human to determine the value of the attribute, such on-the-fly construction can save considerable human intervention by only calling upon them to determine the values when needed.

Effort during policy evaluation can be further reduced by allowing the entity requesting access to provide values for $?$ -defined attributes. The policy evaluation engine can consider these values to be true for the purpose evaluating the policy, but can also store them in a log for later auditing. Those entities providing incorrect values can later be punished by the auditor. Such an approach trades off security for efficiency, a reasonable trade to make in time-sensitive but trustworthy environments. For example, in hospitals, where staff must be able to access needed medical records quickly and are highly trusted and easy to punish through employment actions, such an approach may reasonably manage the risk of staff snooping in unneeded records.

In more detail, let a request be a list of attribute names followed by values:

Request	R	$::=$	$(\text{REQUEST } C_1 \cdots C_n)$
Condition	C	$::=$	$(N V \cdots)$
Attribute Name	N	$::=$	text
Attribute Value	V	$::=$	attribute-value

A request can be computed using the description. For each attribute name N , the description should contain a definition of the form $(N F \cdots)$ where F is a function saying how to compute the values of N . The request R should contain a pair $(N V \cdots)$ where V is equal to the value of F . Multiple values may be listed in the case where N is defined in terms of multiple functions or a function returns multiple values. When F is $?$, the requester's value for R is missing and request must be provisionally evaluated. In this case, the entity making the request can supply purported values for N that can be stored in a log and later be checked during auditing.

An exception to the above method of providing values to attributes comes the basic terms `noninterference` and `differential_privacy` whose semantics are provided by Section 3.2.

Let a request be modeled as a assignment (i.e., a record or dictionary) of attribute names to attribute values. We allow each attribute to be assigned a set of values. Let $\llbracket R \rrbracket[N]$ denote the set of values assigned to the attribute N by the request R . Intuitively, an assignment Σ such that $\Sigma[N] = \{v\}$ is an assertion that the attribute N has the value v . $\Sigma[N] = \{v_1, v_2\}$ is asserting that N has two values, v_1 and v_2 , simultaneously. $\Sigma[N] = \{\}$ is asserting that N has no value, a definite state, and one that is presumably impossible. $\Sigma[N] = \{\top\}$, where $\top$ represents the top most value of attribute value lattice, asserts that the attribute has a value but it is unknown. Thus, a maximally ignorant request is not one that assigns no values to any attributes, but rather one that assigns $\top$ to every attribute, which we will denote as $[\forall \mapsto \{\top\}]$. $\Sigma[N \mapsto \{\cdots\}]$ denotes Σ updated to assign $\{\cdots\}$ to N .

4.4.2 Policy Semantics

Similar to requests, we can treat a target $T = C_1 \cdots C_n$ as such an assignment

$$\llbracket (N \ V_1 \cdots V_m) \ C_2 \cdots C_n \rrbracket = \llbracket C_2 \cdots C_n \rrbracket [N \mapsto \{V_1, \dots, V_m\}] \quad (5)$$

$$\llbracket \cdot \rrbracket = [\forall \mapsto \{\top\}] \quad (6)$$

where $\Sigma[X \mapsto Y]$ denotes adding the assignment of Y to X to an assignment Σ .

We define a relation on assignments similar to subset. Intuitively, the subset-like relation should capture that a request R matches a target $T = C_1 \cdots C_n$, that is, that R is a refinement or, or implies, T . At the level of individual values, a value v_r from a request matches a value v_t from the target iff $v_r \leq v_t$, that is, v_r is a subtype of v_t . When v_r is unknown, that is, $\top$, the request value will only match when the target is also $\top$, either explicitly or implicitly by not being mentioned by the target. At the level of sets of values, a set V_r from the request matches a set V_t from the target iff every request value v_r in V_r has a corresponding target value v_t such that v_r matches v_t . More formally, let $V_r \sqsubseteq V_t$ iff $\forall v_r \in V_r, \exists v_t \in V_t. v_r \leq v_t$. A request value set V_r with $\top$ as a member will only match if V_t also has top as a member. In this case, since a target $\top$ will match any request value, the request will surely match. Finally, let $R \sqsubseteq T$ iff for all attributes N , $R[N] \sqsubseteq T[N]$. The empty target $[\forall \mapsto \{\top\}]$ matches any request.

Here's some examples:

- $\llbracket \cdot \rrbracket \sqsubseteq \llbracket \cdot \rrbracket$ since $R = [\forall \mapsto \{\top\}] \sqsubseteq [\forall \mapsto \{\top\}] = T$.
- $\llbracket (n \ v) \rrbracket \sqsubseteq \llbracket \cdot \rrbracket$ since $R = \llbracket (n \ v) \rrbracket = [\forall \mapsto \{\top\}][n \mapsto \{v\}]$ is such that $R[n] = \{v\}$, $T = \llbracket \cdot \rrbracket = [\forall \mapsto \{\top\}]$ is such that $T[n] = \{\top\}$, and $v \leq \top$.
- $\llbracket (n \ v_1 \ v_2) \rrbracket \sqsubseteq \llbracket \cdot \rrbracket$ since $R = \llbracket (n \ v_1 \ v_2) \rrbracket = [\forall \mapsto \{\top\}][n \mapsto \{v_1, v_2\}]$ is such that $R[n] = \{v_1, v_2\}$, $T = \llbracket \cdot \rrbracket = [\forall \mapsto \{\top\}]$ is such that $T[n] = \{\top\}$, and $v_1 \leq \top$ and $v_2 \leq \top$.
- $\llbracket \cdot \rrbracket \not\sqsubseteq \llbracket (n \ v) \rrbracket$ for $v \neq \top$ since $R = \llbracket \cdot \rrbracket = [\forall \mapsto \{\top\}]$ is such that $R[n] = \{\top\}$, $T = \llbracket (n \ v) \rrbracket = [\forall \mapsto \{\top\}][n \mapsto \{v\}]$ is such that $T[n] = \{v\}$, and $\top \not\leq v$.
- $\llbracket (n \ v) \rrbracket \not\sqsubseteq \llbracket (n \ v') \rrbracket$ where $v > v'$ since $R = \llbracket (n \ v) \rrbracket = [\forall \mapsto \{\top\}][n \mapsto \{v\}]$ is such that $R[n] = \{v\}$, $T = \llbracket (n \ v') \rrbracket = [\forall \mapsto \{\top\}][n \mapsto \{v'\}]$ is such that $T[n] = \{v'\}$, and $v' \not\leq v$.
- $\llbracket (n \ v') \rrbracket \sqsubseteq \llbracket (n \ v) \rrbracket$ where $v' \leq v$ since $R = \llbracket (n \ v') \rrbracket = [\forall \mapsto \{\top\}][n \mapsto \{v'\}]$ is such that $R[n] = \{v'\}$, $T = \llbracket (n \ v) \rrbracket = [\forall \mapsto \{\top\}][n \mapsto \{v\}]$ is such that $T[n] = \{v\}$, and $v' \leq v$.
- $\llbracket (n \ v'_1 \ v'_2) \rrbracket \sqsubseteq \llbracket (n \ v) \rrbracket$ where $v'_1 \leq v$ and $v'_2 \leq v$ since for all v' in $R = \llbracket (n \ v'_1 \ v'_2) \rrbracket$ $v' \leq v$.
- $\llbracket (n \ v' \ v_2) \rrbracket \not\sqsubseteq \llbracket (n \ v) \rrbracket$ where $v' \leq v$ and $v_2 \not\leq v$ since v_2 is in $R = \llbracket (n \ v' \ v_2) \rrbracket$ and $v_2 \not\leq v$. Thus, adding more values to a request can make it no longer match.
- $\llbracket (n \ v' \ v_2) \rrbracket \not\sqsubseteq \llbracket (n \ v \ v_2) \rrbracket$ where $v' \leq v$ and $v_2 \not\leq v$ since $v' \leq v$ and $v_2 \leq v_2$. Thus, adding more values to a target can make it start to match.
- $\llbracket (n \ v_1)(n \ v_2) \rrbracket$ maps n to just v_2 dropping v_1 . This is likely confusing and should be avoided.

A policy rule holds if its target matches the request. The decision of the rule (allow or deny) holds unless one of the exceptions also holds. More formally, let $\llbracket p \rrbracket(r)$ denote the decision yielded by a policy p on a request r . The possible decisions are allow, deny, and na for not applicable.

$\llbracket (\text{ALLOW } t) \rrbracket(r) = \text{allow}$	if $r \sqsubseteq t$
$\llbracket (\text{ALLOW } t) \rrbracket(r) = \text{na}$	if not $r \sqsubseteq t$
$\llbracket (\text{DENY } t) \rrbracket(r) = \text{deny}$	if $r \sqsubseteq t$
$\llbracket (\text{DENY } t) \rrbracket(r) = \text{na}$	if not $r \sqsubseteq t$
$\llbracket (\text{ALLOW } t \text{ EXCEPT } D_1 \cdots D_n) \rrbracket(r) = \text{allow}$	if $r \sqsubseteq t$ and $\llbracket D_i \rrbracket(r) \neq \text{deny}$ for all i
$\llbracket (\text{ALLOW } t \text{ EXCEPT } D_1 \cdots D_n) \rrbracket(r) = \text{deny}$	if $r \sqsubseteq t$ and $\llbracket D_i \rrbracket(r) = \text{deny}$ for some i
$\llbracket (\text{ALLOW } t \text{ EXCEPT } D_1 \cdots D_n) \rrbracket(r) = \text{na}$	if not $r \sqsubseteq t$
$\llbracket (\text{DENY } t \text{ EXCEPT } A_1 \cdots A_n) \rrbracket(r) = \text{deny}$	if $r \sqsubseteq t$ and $\llbracket A_i \rrbracket(r) \neq \text{allow}$ for all i
$\llbracket (\text{DENY } t \text{ EXCEPT } A_1 \cdots A_n) \rrbracket(r) = \text{allow}$	if $r \sqsubseteq t$ and $\llbracket A_i \rrbracket(r) = \text{allow}$ for some i
$\llbracket (\text{DENY } t \text{ EXCEPT } A_1 \cdots A_n) \rrbracket(r) = \text{na}$	if not $r \sqsubseteq t$

Note that while inspired by Legalease, this language differs from it. First, instead of using the subset-like notion of matching given above, for deny rules, Legalease uses an intersection-like notion of matching. Second, Legalease, does not have a notion of not applicable decisions; instead, it would produce allow for deny rules and deny for allow rules when the target does not match.

4.5 Semantics of Origin

The above language does not have an inherent semantics for the attributes and values used in the targets. Instead, the description may, but does not always, provide such a semantics in terms of a function called to determine the value of a term. Central to Origin Privacy are terms defined over the data's origin. The aspects of data's origins that must be captured by an enforcement system depend upon the application. As such, no one mathematical model of origin will be most appropriate for all settings. Nevertheless, to add clarity to our ideas, we provide one such model here.

We model the governed entity as a process with a name E consuming inputs and producing outputs. The state of the process might change over time whereas we presume the name E is held constant. We use e to represent the state of the process at a point in time. One can think of E as a random variable that changes over time and e as a value it takes on. Abusing notation in the standard way, we write $E = e$ to denote that E takes on the value e at the contextually relevant time. While we speak of E as being a process and e being a value, there is a sense in which e is also a process: the state e includes control state, or code, running E and this state can be viewed as a process. This control state, along with inputs and outputs to the entity, determines how the process evolves over time.

We use $\mathbf{e}$ to represent a series of processes at times. (We're assuming a discrete model of time.) We write $\dot{E} = \mathbf{e}$ to mean that the name E refers to a process going through the states $\mathbf{e}$.

The process modeling the system governed by the policy interacts with other entities similarly modeled as a named series of process states.

The inputs and outputs of an entities are messages. We model a message as is a tuple $m = \langle S, R, t, x \rangle$ where S is the entity sending it, R is the entity receiving it, t is the time at which it is sent,

and x is the text of the message. All the inputs to an entity named R must have the form $\langle S, R, t, x \rangle$ for some S , t , and x . All the outputs from an entity named S must have the form $\langle S, R, t, x \rangle$ for some R , t , and x . The outputs of one entity can form the inputs to another. In particular, a message $\langle S, R, t, x \rangle$ is in the proper form to be both an output from S and an input to R . Intuitively, all the messages sent form a directed graph where the entities form the nodes and an entity E_1 sending one of its outputs to an entity E_2 , which consumes it as an input, is represented as a directed edge from E_1 to E_2 .

The governed entity has both intrinsic properties and extrinsic properties. The intrinsic properties are deduced by the entity's model e . Extrinsic properties come from how other entities view the governed entity. For example, talent at medical diagnosis is an intrinsic property whereas a license to practice medicine is an extrinsic property. Properties can change over time as an entity's state changes over time. This is why we distinguish between the process's name E and its value at a time e .

An intrinsic property of particular interest to us is how each entity uses its inputs to cause its outputs. This is a form of *local causation* since the whole story of what causes an output might depend upon other entities as well, making it a combination of intrinsic and extrinsic properties. We suppose that the model includes a relation $\hookrightarrow_E$ for each entity E such that $i \hookrightarrow_E o$ means that the input i caused the entity E to output o , possibly along side of other causes. Such relation could be determined by the model of causation presented in Section 3.2.

We can develop a raised notion of *transitive causation* $\hookrightarrow^*$ that models causation from interacting entities by taking the transitive closure of the union $\hookrightarrow_E$ for all E . For example, suppose that $i_1 \hookrightarrow_{E_1} o_1$ in an entity E_1 and that $i_2 \hookrightarrow_{E_2} o_2$ in entity E_2 . If the output o_1 of E_1 is the input i_2 of E_2 , that is $o_1 = m = i_2$, then $i_1 \hookrightarrow_{E_1} m \hookrightarrow_{E_2} o_2$. In this case, $i_1 \hookrightarrow^* o_2$. Depending upon the notion of causation used, this might not imply that i_1 causes o_2 (that's *causes* in a more global, normal sense of causation than our family $\hookrightarrow_-$ of local causal relations). For example, the model of causation presented in Section 3.2, is not transitive.

We use transitive causation instead of normal causation for reasons of efficiency. Computing the transitive closure of each entity's causation relation is much easier than computing a global causation relation. Furthermore, the two notions only differ in cases where two different causal paths cancel out. Such situations tend to be unstable and so unnatural that arguably transitive causation captures our intuitions about information flow better than causation itself.

The origin of an output o given such a model is, roughly speaking, all the messages m such that $m \hookrightarrow^* o$ along with the entities that produced them labeled with all the properties they had at the time of producing the messages. This can be viewed, roughly, as a subset graph of the graph of all entities where the edges are determined by messages, as discussed above. More precisely, the sub-graph must also be labeled the properties of those involved entities at the times of producing the relevant messages. We call this the *history* of the output o .

Policies written to purely use only Origin Privacy determine the permissibility of an output o solely based the history of o .

The history of the o is open-ended in the sense that we could always include parts of history further back in time. In practice, policies should be closed-ended in the sense of referencing only history of a fixed amount in the past.

Further complicating matters, the history of an output o might not be directly observable. Rather, an enforcement mechanism might only have access to approximation of the output's history, which we call a *log*. For an Origin Privacy policy to be enforceable, it should only refer to

parts of an output’s history that can be determined from the log files to which enforcement mechanisms have access.

In more detail, we model the interacting entities using a trace-like model. Let $\mathcal{E}$ be the set of all entity names. Let $\mathcal{P}$ be the set of all property names. Let a *local status* s be a function from $\mathcal{P}$ to the value of the property for some entity. Let a *global status* g be a function from $\mathcal{E}$ to the local status of the supplied entity. Let a *history* h be a function from $\mathcal{N}$ to a global status. $h(0)$ is the current global status; $h(t)$ is the global status t time units ago, or at time $-t$ using the current time as the epoch. A *log* is similar to a history, but created out of partial local statuses (a partial function) with undefined cases implying a lack of knowledge.

One class of properties record the messages an entity send at some point time; another record those messages received. Thus, the history records all the messages sent and received. The attributes of a policy can include references to these messages.

4.6 Implementation Prototype

We have implemented a label-tracking system to enforce origin policies in (Scala) programs. The system is based on both static types and dynamic label tracking: computations and data have to be statically typed to indicate that they either process or are sensitive information but the particulars of origin (or other labels) are tracked dynamically. The static component of the system relies on the standard Scala type system. The implementation allows the creation and enforcement of Legalese-like policies. Policies are evaluated dynamically.

The implementation allows simple interoperation with standard Scala libraries for big data like Apache Spark. A demo illustrates a simple map+reduce task written using Spark primitives that can be distributed among a Spark computing cluster while maintaining origin labels throughout its evaluation.

The implementation, its documentation, and demo programs are available in the code bundle, which we have sent to DARPA [22], and at <https://github.com/cmu-transparency/demo-scala-labels>.

The prototype is a combination of two main components: a label tracking system and a label-based policy specification and enforcement mechanism. Together these components allow us to track data-attached concepts such as origin throughout the execution of a program, and enforce a policy specified in terms of those concepts before the program is allowed to exfiltrate its results. We provide example policies demonstrating origins, purposes, and roles as described in Section 4.3, and implementing the semantics described in Section 4.4.

The prototype is more general than the policies and concepts defined earlier in this document. However, it constitutes a shallow embedding of both label tracking and the policy language within the Scala language, in that it does not require any modifications to its host language, its syntax, or its semantics. The label tracking component relies on programs being written in monadic-style, making use of the provided monads, which interject label tracking within the program’s normal operation. Likewise, constructing policies relies on standard Scala syntax, deviating from the syntax presented in Section 4.3, but can nevertheless be used to describe the same policies.

It is helpful for the reader wishing to follow along with the rest of this section to be familiar with the basics of Scala and its syntax. An introduction to the language can be found at <https://docs.scala-lang.org/tour/basics.html>. We briefly describe the label tracking component next but focus on the policy declarations and their enforcement for the rest of this

section.

4.6.1 Attributes as Labels

Three features underlie our implementation of data attributes: *labels*, *labeled data*, and *label-sensitive computations*. Attributes and their values as described in Section 4.4 are modeled as labels (the term derives from language-based security literature). Labels are arbitrary annotations (with some requirements described later) that accompany data throughout a program computation. Data can be *labeled* with a label and stored in that form on an outside system such as a database. Finally, label-sensitive *computations* are monadic-style programs that are allowed to read labeled data, but can only be executed if allowed by a policy.

Example labels in the prototype implementation include labels for tracking origins like persons (users), locations, and times, and contextual information such as purposes and roles. For technical reasons, each label is an approximation of a set of attribute values, in that a labeled data element is annotated with an over-approximation and under-approximation of the potentially large set of values that can be “attributed” to the data as computed. For this reason, labels are required to come with a least upper bound ($\sqcup$) and greatest lower bound ($\sqcap$) operations, as well as special top ($\top$) and bottom ($\perp$) elements representing approximations of all or no values of a given attribute, respectively.

The intended use case of the labeling system is as follows.

- Sensitive information stored in a database or arising from other sources comes into a program as data of type *Labeled*[L, T] where L is the label type and T is the underlying data type. For the prototype demonstration, the label type is fixed and does vary.

```
val secret: Labeled[L, String] = ...
```

- Computations that access labeled data are written in the *LIO*[L] monad. Computations that eventually produce a value of type T are thus typed *LIO*[L, T].

```
val ready: LIO[L, Boolean] = for {  
  actual_secret <- unlabel secret  
} yield (actual_secret == "attack_at_dawn")
```

Within this computation, labeled data can be unlabeled at will but doing so necessitates considering that data’s label at the point that the computation’s output is retrieved.

- Finally, retrieving a piece of data computed with the *LIO*[L] monad, it is necessary to provide a policy governing the program.

```
val actually_ready: Boolean =  
  ready.evalTCB(context, policy)
```

The `evalTCB` method executes the *LIO*[L] monad, integrates the context (labels such as role and purpose) with the label attached to the result of the computation (together constituting the *request* as in Section 4.4), and evaluates the given policy on the constructed request.

The result is either the unlabeled version of the computed data which is assumed to be immediately used for its intended purpose. The “TCB” here refers to “trusted computing

base”, indicating source code areas that are presumed to be trusted and thus require special scrutiny on behalf of the programmer. If the policy fails to allow the request, an exception is raised instead.

Some utility classes are provided in the prototype and can save the programmer some work. For example, for cases of attributes that can only exhibit a small number of different values of type *T*, the *USet*[*T*] class implements a label that is exact for that attribute, in that it tracks exactly all of the values for that attribute that arise in the execution of some computation. This is used in the demo to define labels for (exactly) representing the purpose and role attributes and requires little effort on behalf of the programmer:

```
object Purpose extends ... USet[String] {  
  type T = USet[String]  
  ...  
  object Nothing      extends NoneSet  
  object Everything extends AllSet  
  
  val bot: T = Nothing.asInstanceOf[T]  
  val top: T = Everything.asInstanceOf[T]  
  
  val Legal          = Purpose("Legal")  
  val ClimateControl = Purpose("ClimateControl")  
  val Sharing        = Purpose("Sharing")  
}
```

The *USet* in the above implements the join and meet operations necessary for labels. In this case they are merely exact set union and set intersection.

The labeling approach in the prototype is loosely based on the *Labeled IO* (LIO) library for Haskell. It is available at <https://github.com/plsyssec/lho>. The rest of this section focuses on the policy enforcement component. For further details regarding the label tracking system, please consult its introductory documentation at <https://github.com/cmu-transparency/demo-scala-labels>. The implementation is available at the same address.

4.6.2 Policies

Policies indicate the conditions under which data protected by the *LIO* monad can become available outside of it. Evaluation of a protected computation and the policy check is performed using the *TCBeval* operation on an *LIO*[*L*, *T*] computation.

```
def TCBeval(context: Label, policy: Policy): T
```

The method returns a value of type *T* or fails with a policy violation if the given policy does not allow the request composed of the labels of the data arising from this monad’s computation, and the additional context label which conveys contextual annotations such as purpose and role.

Policies are composed of a basic components which, most generally, are functions that, given a request label, return a value indicating an allowed release (*Some*(*True*)), denied release (*Some*(*False*)), or no decision (*None*).

```

trait Policy ... {
  def apply(request: Label): Option[Boolean] = ...
}

```

4.6.3 Conditions

As noted, the above definition of a policy is very general and were it used in its full generality, would result in unreadable and difficult to understand policies. Instead, we provide a collection of functions and constructors that are intended to produce conditions on origins, roles, and aspects like those described earlier in this document.

The design of the conditions relies on comparison operations between labels (attributes), using one of the following label functions:

```

def  $\sqsubseteq$ (a: Label, b: Label): Boolean = a  $\sqcup$  b == b
def  $\sqsupseteq$ (a: Label, b: Label): Boolean = a  $\sqcap$  b == b
def  $\sqsubset$ (a: Label, b: Label): Boolean = a  $\sqsubseteq$  b && a != b
def  $\sqsupset$ (a: Label, b: Label): Boolean = a  $\sqsupseteq$  b && a != b

```

Note that the equality comparison is not provided here as labels are over/under-approximations of sets of attributes.

Various implicit definitions are provided in the prototype that allow the user to define such comparison conditions where the left/a side of the comparison is assumed to be a component of the eventual request, and the right/b side is a particular given label. These are analogous to attributes and attribute values as in Section 4.4. For example, the condition (*Origin* users) can be written in Scala:

```
Origin.Person  $\sqsubset$  Origin.Person.bot
```

Here *Origin.Person.bot* denotes the bottom element (least element) in the lattice representing user origins. The comparison here says that the condition holds when a request's user origin is strictly greater than bottom, or in other words, originating from at least one user.

Further utility methods allow us to combine the basic conditions such as in:

```
Origin.Person  $\sqsubset$  Origin.Person.bot
and Purpose  $\sqsupseteq$  Purpose.Sharing
```

In the second part of the condition above we require that the request's purpose includes at least the Sharing purpose. This condition models the first DENY clause in the example policy of Section 4.3. The reader might have noticed that the conditions specified in our prototype are more complicated than the ones presented in the syntax sections earlier in this paper. Indeed, the prototype requires some understanding of the underlying label approximation system in order to pick the right comparison operator and the right operands for the condition being modeled. This is a shortcoming of the prototype but we conjecture that it could be side-stepped in future implementations.

4.6.4 Legalese-like Policies

Base policies can be combined into larger compound policies that are more convenient at specifying a real-world policy. The prototype implements Legalese-like policies as described in Sec-

tion 4.3. The syntax differs from the one presented there given the implementation restricts itself to a shallow embedding into the Scala host language. We have, however, provided several implicit definitions and classes that aid in writing policies in a manner similar to the one described in Section 4.3. A policy like the one exemplified in Figure 2 is written in Scala below:

```
allow.except (
  deny(Origin.Person  $\sqsupseteq$  Origin.Person.bot
    and Purpose  $\sqsupseteq$  Purpose.Sharing)
    .except(Seq(
      allow(Role  $\sqsupseteq$  Role.Affiliate),
      allow(Purpose  $\sqsupseteq$  Purpose.Legal)
    ))
)
```

One of the inner allow clauses is missing in the above as we have not fully modeled all of the attributes necessary for that example. The prototype does not directly provide the attribute specification language show in Figure 5. The label approximations of attributes necessary for the system to model any given scenario need to be implemented by the programmer. Though this is tedious work, we envision improvements and specification-language to Scala translations are possible.

4.6.5 Demo: Big Data Processing

One of the advantages of a *shallow embedding* of label tracking and policy enforcement is that it can be used with the host language’s libraries with little additional effort. This makes it easy to keep data with annotated attribute values in databases, files, and other storage mechanisms. Likewise, labeled values can be used in parallel or big data workflows such as Apache Spark. The prototype includes a demo use case exercising both serialization and big data library use. We summarize some elements of this demo in this section.

First, the demo defines several types of labels, collectively organized into the class *DemoLabel*:

```
class DemoLabel(
  val person: Origin.Person      = Origin.Person.bot,
  val location: Origin.Location  = Origin.Location.bot,
  val time: Origin.Time          = Origin.Time.bot,
  val purpose: Purpose           = Purpose.bot,
  val role: Role                 = Role.bot)
...
extends Tuple5[...]
with Label[DemoLabel]
with Serializable ...
```

For brevity in the following code listing, we make use of several type aliases:

```
type DL = DemoLabel
type Ld[T <: Serializable] = Core.Labeled[DL, T]
type LIO[T] = Core.LIO[DL, T]
```

Demo Overview The Demo is a simplified model of the WiFi device tracking present in Brenn Hall at the University of California, Irvine. The basic data involved in this model includes persons, locations, sensors, and sensor readings:

```
object DT { // for ``demo types``
  type Id = Long
  type Signal = Double
  type Name = String

  case class Person(val device_id: Id, val name: Name)
  case class Location(val location_id: Id, val name: Name)
  case class Sensor(val sensor_id: Id, val location: Location)
  case class SensorReading(
    val sensor_id: Id,
    val signal_strength: Signal,
    val timestamp: Time
  )
}
```

The *Person* and *Location* types also function as attribute values for the purpose of origin tracking. Additionally, two more types are solely for use in the enforcement scheme, purposes, and roles:

```
case class Purpose(val name: Name)
case class Role(val name: Name)
```

This data models a set of WiFi sensors and people with WiFi devices. Each person is associated with a particular device and each sensor reading indicates the detected device. Device ids are not themselves sensitive but the mapping from device id to a person is. Likewise, sensor ids are not sensitive but their locations are. To model this sensitivity, we employ origin labels. For the demo we populate a fictional collection of data points of both types and label them according to their origins. We envision this process would be done by an administrator and simulate it in for the demo via a hard-coded roles and purposes, and a policy allowing the labeling of data only under the specific hard coded role and purpose:

```
private object TCB_PopulateData {
  ...
  val dataIngressPolicy = deny.except(Seq(
    allow(Role  $\sqsupseteq$  Role.Administrator
      and Purpose  $\sqsupseteq$  Purpose.Storage)
  ))
  val simulatedContext: DL =
    (Purpose.Storage: DL)  $\sqcup$  (Role.Administrator: DL)
    // implicitly converted to DemoLabel
}
```

The labeled data is thus constructed from raw unlabeled data, by way a computation performing a label operation, and the evaluation of the said policy-governed computation via TCBeval:

```
val rawUsers: Seq[DT.Person] = Seq(
  DT.Person(1001, "Piotr_Mardziel"),
```

```

    DT.Person(1002, "Anupam_Datta"),
    DT.Person(1003, "Michael_Tschantz"),
    DT.Person(1004, "Sebastian_Benthall"),
    DT.Person(1005, "Helen_Nissenbaum")
)
val labeledUsers: Map[DT.Id, Ld[DT.Person]] =
    rawUsers.map { u =>
        (u.device_id, Core.label(u: L, u).TCBeval(
            simulatedContext, dataIngressPolicy
        )) }.toMap

```

Notice that each person is labeled with an origin where the attribute value is the per itself. The scheme re-uses persons for both data and labeling. A similar process is used to label each sensor, producing:

```

val labeledSensors: Map[DT.Id, Ld[DT.Sensor]] = ...

```

The labeled data thus constructed is only accessible from a protected computation executed under a policy and a context. The data population step above had access to raw unlabeled data as a simulation of the process used by an administrator to add sensitive data to a system.

External Storage The data constructed in the prior section can easily be stored in a database or some other system. As an example, we employ Java's standard serialization scheme (here this is performed by the method `write`) to store it in plain files:

```

private object TCB_PopulateData {
    ...
    write("users.data", labeledUsers)
    write("sensors.data", labeledSensors)
    ...
}

```

Further system elements then rely on the pre-constructed stored data sets, which when loaded, come with the various labels attached to them at the time of storage.

```

lazy val users: Map[DT.Id, Ld[DT.Person]] = load("users.data")
lazy val sensors: Map[DT.Id, Ld[DT.Sensor]] = load("sensors.data")

```

Note that this shallow embedding makes it easy to store labeled data externally, it does not make it immediately possible to process the said data using external systems. For example, storing binary Serialized data in an SQL database makes said data retrievable, but it does not allow for inspection of the data within SQL queries.

Big Data Processing via Spark Labeled data or labeled computations can also be stored and processed in a Spark cluster, using its resilient distributed data-set capabilities (RDD). Spark's RDDs are quite general and support various parallel programming paradigms, including map-reduce like computations which we demonstrate below. More information about RDDs and capabilities of Spark in general can be found at <https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd.html>.

If the set of sensor readings is expected to be large, we can store them in an *RDD*:

```
val readings: RDD[DT.SensorReading] =  
  SparkUtil.rdd(load("readings.data"))
```

This itself is not yet labeled data but will become labeled in the example that follows. Further, in an actual deployed system, this data would not be stored in a single file but distributed to begin with via standard distributed file systems like HDFS. Alternatively the data could be processed as a stream, the the labeling code above employed as the stream's first pre-processing step.

Once in the form of an *RDD*, data processing can take advantage of parallel processing allowed by Spark. For example, the method `occupancy`, which computes the number of readings present at a given location, can employ the eponymous *map*, *reduce* sequence for its task:

```
def locate_sensor(sensor_id: DT.Id): LIO[DT.Location] = ...  
  
def occupancy(  
  readings: RDD[DT.SensorReading],  
  location: DT.Location): LIO[Int] = {  
  
  map(readings) {  
    case reading => for {  
      loc <- locate_sensor(reading.sensor_id)  
    } yield if (loc == location) { 1 } else { 0 }  
  }.reduce(implicitly[Numeric[LIO[Int]]].plus)  
}
```

Note that the output of the `map` operations is of type `RDD[LIO[Int]]`, that is, a distributed collection of label manipulating computations. The `map` operation is performed in parallel across a cluster of workers, and the following `reduce` operation coalesces the variously distributed values into a single end result.

4.7 Bayesian networks and information flow security

In this section, we formalize the ontology from Section 4.2 that we derived from privacy policies. Our formalization uses the causal graphical modeling tools outlined in Section 3.4. We show that several known results in information flow security have dual results within our formal causal modeling of systems in their environments. We demonstrate this for concepts of noninterference [6] and the impossibility of guaranteeing secrecy given the possibility of arbitrary auxiliary knowledge [7, 27].

Where our analysis goes beyond these known results in information flow security are that our models explicitly take into account the relationship between a technical system and its environment. In each case, our theorems prove conditions of the security properties that could not be discovered by static program analysis in isolation. For example, it is well known that information can be revealing of other sensitive information given auxiliary knowledge. Our Theorem 36 reflects the conditions under which auxiliary knowledge is possible.

Because the CPD defined by Pr between each random variable and its parents can be an arbitrary function, including deterministic logical operations, it is possible to encode a system of

computational components, including sensors, data processors, and actuators, as a BN. Earlier we defined Origin Privacy in terms of systems, processes, and messages. These concepts map easily into this formalism: systems are Bayesian networks; processes are random variables or events; the inputs and outputs of processes are determined by links connecting them to other processes; messages are the instantiation of particular random variables, which are available as inputs to later variables.

4.7.1 Embedded Causal System (ECS) Model

In the subsequent sections we will use the following standard notation and model, which we will refer to as the Embedded Causal System (ECS) Model.

Definition 15 (World). A *world* $\mathcal{W}$ is a set of random variables with a conditional probability distribution that may be modeled as a Bayesian network $(G_{\mathcal{W}}, Pr_{\mathcal{W}})$.

$\mathcal{W}$

Definition 16 (System). A subset of the world $\mathcal{Y} \subset \mathcal{W}$ is the *system*.

Definition 17 (Environment). The *environment* of $\mathcal{Y}$ is the set of nodes in the world that are not in the system, $\mathcal{E} = \mathcal{W} - \mathcal{Y}$

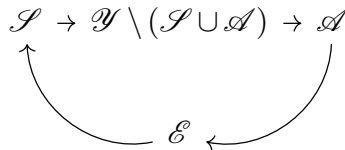


(In this and a few other diagrams, we will include cycles because these are diagrams of blockmodels over other networks. In a blockmodel of G , a partition $\{P_1, P_2, \dots\}$ of the set of original nodes $\mathcal{X}$ is treated as a new set of nodes, with an edge between P_1 and P_2 iff there exists X_1 and X_2 such that X_1 is in P_1 , X_2 is in P_2 , and (X_1, X_2) is in G_{edges})

It is common in security research to consider *systems* as units of analysis; these systems contain *programs*; system input is data and system output is the result of the programs operating on the data [33]. These programs are represented using a formal approximation of a programming language in order to prove security properties of systems. Systems in our formalism also have inputs and outputs, which are defined by their position relative to the environment.

Definition 18 (Sensors and inputs). A *sensor* is an edge $(A, B) \in G_{\mathcal{W}}$ such that $A \in \mathcal{E}$ and $B \in \mathcal{Y}$, An *input* is the head node of a sensor, B . Denote the set of all inputs with $\mathcal{I}$.

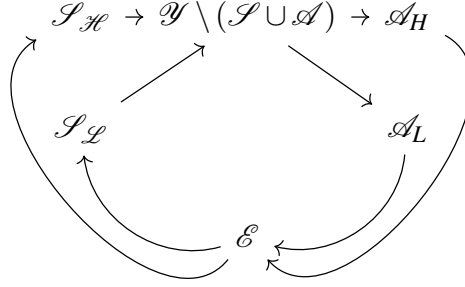
Definition 19 (Actuators and outputs). An *actuator* is an edge $(A, B) \in G_{\mathcal{W}}$ such that $A \in \mathcal{Y}$ and $B \in \mathcal{E}$. An *output* is the tail node of an actuator, A . Denote the set of all outputs with $\mathcal{A}$.



See the definition of *orderly* (Definition 25) for a set of further constraints on inputs and outputs that are necessary for proving security properties of ECS models.

For some security related applications of this model, it is necessary to distinguish between “high-side” and “low-side” inputs and outputs. High-side variables denote sensitive variables that should not be leaked.

Definition 20 (High and low sides). Inputs $\mathcal{I}$ are partitioned into high $\mathcal{I}_H$ and low $\mathcal{I}_L$ side variables. Similarly, outputs $\mathcal{A}$ are each partitioned high-side $\mathcal{A}_H$ and low-side $\mathcal{A}_L$ variables.



Note that in the above diagram and throughout this paper, we will sometimes refer to a set of random variables such as the set of all high-side inputs $\mathcal{I}_H$ as if it is a single random variable. This is well-motivated, because for any set of random variables $\mathcal{X} = \{X_0, X_1, X_2, \dots\}$ one can define a new random variable whose domain $\text{Dom}(\mathcal{X})$ is the cross product $\text{Dom}X_0 \times \text{Dom}X_1 \times \dots$ and whose probability distribution is the joint probability distribution $\text{Pr}(X_0, X_1, \dots)$.

Example 21. A hospital uses a system to manage its medical records. It takes input from many health care professionals through many different forms of treatment. Most medical records are considered a low-side input because they can be accessed by other professionals treating the patient. Psychotherapy notes are a high-side input because they have special restrictions on their use.

Example 22. An intelligence agency has many classified sensors, such as satellites and drone imagery, which contain information that is critical for national security. These are high-side inputs. They also use many data sets that are available publicly and commercially. These are low-side inputs.

4.7.2 System design

In many cases what we are interested in is the possibility of a *system designer* inventing a system subject to certain constraints.

We are interested in the ways that an ECS enables inferences, and how these inferences depend on what is known or observable about the system and its environment. We define some terms here to denote properties of the conditioning set that we will use in later proofs. Intuitively, conditioning sets can be interpreted as observed states of the world that are available to an adversary trying to learn high-side information.

The condition of being *present* captures the intuition that system designers cannot account for the ways that downstream uses of system outputs may be used to reveal sensitive information.

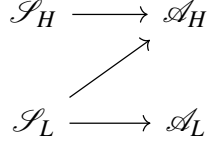


Figure 6: A system that is safe so long as its high-side actuators are not observed

Definition 23 (Present). A system $\mathcal{Y}$ with conditioning set $\mathcal{C}$ is *present* iff

- No descendants of $\mathcal{A}$ are in the conditioning set $\mathcal{C}$, and
- No descendant of $\mathcal{A}$ is in $\mathcal{S}$.

The term “present” indicates that the attacker is able to condition on variables prior to and during the operation of the system, but not variables in the “future” of the system. Requiring that the system outputs are not ancestors of the the system inputs guarantees that the system is in fact positioned in a particular place in time, so to speak.

The condition of being *covered* captures the intuition that in general we do not expect attackers to have the ability to observe systems *as they are functioning*, even if we allow them to know exactly how a system works because they know the causal relationships between the system components.

Definition 24 (Covered). A system is *covered* if no $Y \in \mathcal{Y}$ is in the conditioning set $\mathcal{C}$.

We also specify a condition on the relationship between sensors, actuators, the system, and the environment. In some cases we will not allow an input to be caused by a system variable. We will also not allow an output to cause a system variable. When both these conditions hold, we call a system *orderly*.

Definition 25 (Orderly). A system is *orderly* iff:

- $\forall X \in \mathcal{W}, S \in \mathcal{S}, X \in Pa(S) \implies X \in \mathcal{C}$
- $\forall X \in \mathcal{W}, A \in \mathcal{A}, A \in Pa(X) \implies X \in \mathcal{C}$

Given only the subgraph represented by $\mathcal{Y}$, under some condition it will be the case that the high-side inputs and the low-side outputs are conditionally independent. We will name this property *safety*.

Definition 26 (Safe). A system $\mathcal{Y}$ is *safe* given conditioning set $\mathcal{C}$ iff when considering it as a subgraph, there are no unblocked paths between $\mathcal{A}_L$ and $\mathcal{S}_H$.

If there are no unblocked paths between $\mathcal{A}_L$ and $\mathcal{S}_H$ in the system subgraph, then these variables are d-separated and so the system is safe. We assume for our purposes that a system designer can guarantee its safety through sound engineering alone.

For example, consider the system defined by the graph in Figure 4.3.

When $\mathcal{A}_H$ is not in the conditioning set, the path between $\mathcal{S}_H$ and $\mathcal{A}_L$ is blocked, so the high-side input and low-side outputs are independent.

In general, system designers can guarantee safety by removing any direct paths from $\mathcal{S}_H$ to $\mathcal{A}_L$ and then ensuring that the system is covered.

Theorem 27. If a system subgraph $\mathcal{Y}$ has no direct paths from $\mathcal{S}_H$ to $\mathcal{A}_L$ and is covered and orderly, then it is safe.

Proof. Assume there are no direct paths from $\mathcal{S}_H$ to $\mathcal{A}_L$ in the system subgraph. So any unblocked path between $\mathcal{S}_H$ and $\mathcal{A}_L$ must be indirect.

Suppose there is a path p that begins with an incoming edge to $\mathcal{S}_H$, as in $\mathcal{S}_H \leftarrow \dots \mathcal{A}_L$. Because the system is orderly, incoming edges to input $\mathcal{S}_H$ must come from nodes that are not in the system $\mathcal{Y}$, as in $\mathcal{S}_H \leftarrow \mathcal{E} \dots \mathcal{A}_L$. Because these nodes are not in the system subgraph, they cannot be in p .

Therefore, the path p must begin with an outgoing edge to $\mathcal{S}_H$. By a parallel argument, the path must end with an incoming edge to $\mathcal{A}_L$, as in $\mathcal{S}_H \rightarrow \dots \rightarrow \mathcal{A}_L$.

Because the path p is indirect and it begins with an outgoing edge and ends with an incoming edge, there must be some node X such that X is on the path and X is a common effect node, as in $\mathcal{S}_H \rightarrow \dots \rightarrow X \leftarrow \dots \rightarrow \mathcal{A}_L$.

Because the system is covered, X must be unobserved. This implies that the path p is blocked. Therefore, there are no unblocked paths between $\mathcal{S}_H$ and $\mathcal{A}_L$. Thus, by Theorem 11, these variables are independent and the system is safe. $\square$

Information flow security literature often considers systems or programs in isolation from their environment. In practice, systems are always connected with an environment, which is why we have developed ECS. So Theorem 27 is not enough to show the conditions of security in an ECS model because its inputs and outputs are not connected with variables in an environment. For this, we turn to a well established formal security model, noninterference.

4.7.3 Noninterference

Noninterference, introduced by [6], is a security policy model widely used in computer science. Sabelfeld and Myers [57] define noninterference informally as “a variation of confidential (high) input does not cause a variation of public (low) output.”

More formally, model a program C as taking an input state $s = (s_h, s_l)$, as producing an output in a set $S \cup \{\perp\}$, where $\perp$ stands for nontermination and $\perp \notin S$. We use $\llbracket C \rrbracket : S \rightarrow S \cup \{\perp\}$ to denote such a semantics. An equivalence relation $=_L$ holds when two inputs they agree on the low values ($s =_L s'$ iff $s_l = s'_l$). The attacker’s power is characterized by a relation $\approx_L$ such that if two behaviors are related by $\approx_L$ they are indistinguishable to an attacker.

Following Tschantz et al. [29] and Datta et al. [19], we expand the definition of the operator $\llbracket \cdot \rrbracket$ to be a function to and from probability distributions over states, which affords a probabilistic definition of noninterference.

Definition 28 ((Probabilistic) noninterference). For a given semantic model, C exhibits *noninterference* or is *secure* iff for all s_1 and s_2 in S , $s_1 =_L s_2$ implies $\llbracket C \rrbracket(s_1) \approx_L \llbracket C \rrbracket(s_2)$.

This definition admits a wide range of possible semantics for the attacker’s equivalence relation $\approx_L$.

We will choose a particular semantics relevant to the ECS model. We impose a probability distribution over inputs $\mathcal{S}$. With it we can construct the variable $\mathcal{A} = \llbracket C \rrbracket(\mathcal{S})$. We use $\mathcal{Y}$ to denote the minimal Bayesian network relating $\mathcal{S}$ to $\mathcal{A}$ and treat it as the system model. As per the

ECS model, we partition the inputs and outputs into high and low sides, $(\mathcal{S}_H, \mathcal{S}_L)$ and $(\mathcal{A}_H, \mathcal{A}_L)$, respectively. Define attacker indistinguishability $\approx_L$ as probabilistic indistinguishability of the low-side outputs when conditioned on inputs:

Definition 29 (ECS attacker indistinguishability).

$$\mathcal{A} \approx_L \mathcal{A}' \text{ iff } Pr(\mathcal{A}_L) = Pr(\mathcal{A}'_L)$$

Conceptually, we are modeling the execution of the program C as the realization of the random variables in $\mathcal{Y}$. C implies a probability distribution $Pr(\mathcal{A}|\mathcal{S})$. It also implies a probability distribution of $\mathcal{A}$ conditional on $\mathcal{S} = s$ realized. $\llbracket C \rrbracket(s) = Pr(\mathcal{A}|\mathcal{S} = s)$, where s is an instantiation of $\mathcal{S}$. For $s \in \mathcal{S}, a \in \mathcal{A}$, let $(s_h, s_l, a_h, a_l) =_L (s'_h, s'_l, a'_h, a'_l)$ iff $s_l = s'_l$.

Definition 30 (ECS Noninterference). For a given ECS model, $\mathcal{Y}$ exhibits *noninterference* or is *secure* iff

$$\forall s_1, s_2 \in \mathcal{S}, s_1 =_L s_2 \implies P(\mathcal{A}|\mathcal{S} = s_1) \approx_L P(\mathcal{A}|\mathcal{S} = s_2)$$

Corollary 31. $\mathcal{Y}$ is secure by noninterference iff

$$\mathcal{A}_L \perp\!\!\!\perp \mathcal{S}_H \mid \mathcal{S}_L$$

Proof. $\mathcal{Y}$ is secure by noninterference iff

$$\forall s_1, s_2 \in \mathcal{S}, s_1 =_L s_2 \implies P(\mathcal{A}|\mathcal{S} = s_1) \approx_L P(\mathcal{A}|\mathcal{S} = s_2)$$

iff

$$\begin{aligned} & \forall s_H, s'_H, s_L, s'_L \in \mathcal{S}, s_L = s'_L \\ & \implies P(\mathcal{A}|\mathcal{S}_H = s_H, \mathcal{S}_L = s_L) \approx_L P(\mathcal{A}|\mathcal{S}_H = s'_H, \mathcal{S}_L = s'_L) \end{aligned} \quad (7)$$

iff

$$\begin{aligned} & \forall s_H, s'_H, s_L \in \mathcal{S} \\ & P(\mathcal{A}_L|\mathcal{S}_H = s_H, \mathcal{S}_L = s_L) = P(\mathcal{A}_L|\mathcal{S}_H = s'_H, \mathcal{S}_L = s_L) \end{aligned} \quad (8)$$

iff

$$\mathcal{A}_L \perp\!\!\!\perp \mathcal{S}_H \mid \mathcal{S}_L$$

□

Under what conditions is a system secure by noninterference?

We can prove that if the system designer can guarantee that the system is present and safe, then it is secure by noninterference.

Lemma 32. If an ECS model is present, then there can be no unblocked path between $\mathcal{S}_H$ and $\mathcal{A}_L$ that includes an outgoing edge from $\mathcal{A}_L$.

Proof. Proof by contradiction.

Suppose an unblocked path exists between $\mathcal{S}_H$ and $\mathcal{A}_L$ such that the edge connecting to $\mathcal{A}_L$ was outgoing. That is, suppose the path was of the form:

$$S_H \cdots \leftarrow \mathcal{A}_L$$

for some S_H in $\mathcal{S}_H$.

Consider the sequence of nodes on the path $S_H, X_1, X_2, \dots, X_n, \mathcal{A}_L$ and the direction of the arrows between them, with $X_n \rightarrow \mathcal{A}_L$, and labeling S_H with X_0 .

Because the system is present, no descendant of $\mathcal{A}_L$ is in $\mathcal{S}$. Therefore there must be at least one edge on the path such that $X_{i-1} \rightarrow X_i$.

Count down from n to 1 and identify the first X_i such that $X_{i-1} \rightarrow X_i$,

X_i will be a descendant of $\mathcal{A}_L$ because there is a direct path between $\mathcal{A}_L$ and it. X_i cannot be $X_0 = S_H$, which is in $\mathcal{S}$.

Therefore node X_i will be the common cause of a head-to-head connection. That is,

$$S_H \cdots X_{i-1} \rightarrow X_i \leftarrow X_{i+1} \leftarrow \cdots \leftarrow \mathcal{A}_L$$

Because the system is present, this node X_i must not be in the conditioning set. The path must therefore have a head-to-head connecting node that is not in the conditioning set. So it is a blocked path, resulting in a contradiction. $\square$

Theorem 33. Given an ECS model, if the system is present, safe, and orderly then the system is secure by noninterference.

Proof. Consider possible paths between $\mathcal{A}_L$ and $\mathcal{S}_H$ while $\mathcal{S}_L$ is in the conditioning set.

By Lemma 32, a present system can have no unblocked paths from $\mathcal{S}_H$ to $\mathcal{A}_L$ that end with an outgoing edge from $\mathcal{A}_L$. So any unblocked path from $\mathcal{S}_H$ to $\mathcal{A}_L$ must end with an incoming edge into $\mathcal{A}_L$.

Because the system is orderly, if $X \rightarrow A$ for any A in $\mathcal{A}$, then $X \in \mathcal{Y}$. Therefore, unblocked paths must include nodes in the $\mathcal{Y}$ subgraph.

Because the system is safe, there are no unblocked paths between $\mathcal{S}_H$ and $\mathcal{A}_L$ consisting of only nodes in the system $\mathcal{Y}$ subgraph.

So any unblocked path between $\mathcal{S}_H$ and $\mathcal{A}_L$ must include both nodes that are in $\mathcal{Y}$ and nodes that are in $\mathcal{E}$. We have already ruled out paths that include descendants of $\mathcal{A}$. So such a path must include ancestors of $\mathcal{S}$. The path must begin with $\mathcal{S}_H$, go into the environment, then re-enter the system via $\mathcal{S}_L$, then go to $\mathcal{A}_L$. Because the system is orderly, incoming edges into $\mathcal{S}_L$ must be $E \in \mathcal{E}$ and outgoing edges must be $Y \in \mathcal{Y}$. That is,

$$S_H \leftarrow \cdots E \rightarrow S_L \rightarrow Y \cdots \leftarrow \mathcal{A}_L$$

for some S_H in $\mathcal{S}_H$, S_L in $\mathcal{S}_L$, and $\mathcal{A}_L$ in $\mathcal{A}_L$.

S_L is in the conditioning set and part of a head-to-tail structure $E \rightarrow S_L \rightarrow Y$ on the path. Therefore the path is blocked.

So with $\mathcal{S}_L$ in the conditioning set, all paths between $\mathcal{A}_L$ and $\mathcal{S}_H$ must be blocked.

So $\mathcal{A}_L \perp\!\!\!\perp \mathcal{S}_H | \mathcal{S}_L$. By Corollary 31, the system is secure by noninterference. $\square$

It is therefore possible to implement an ECS that is secure in the sense of noninterference as long as a few conditions of the system (covered, safe, and orderly) are guaranteed. Privacy policies that restrict information flow (e.g. by guaranteeing confidentiality) of data based on how it was inputted into the system can be modeled in this framework. In the next section, we will show that policies that impose restrictions on information flow based on the content of information cannot be as easily restricted; to be effective there must be independence relations in the environment of the system. Thus the viability of privacy policies depends on the distinction noted in Section 3.5 between causal flow and association.

4.7.4 Preventing associations

We have proven that under certain conditions an ECS is secure by noninterference (Theorem 33). Noninterference is a widely respected formal security model in among computer security researchers. One reason for this is that it is a criterion that depends only on the internals of the system. Computer scientists can guarantee that a system, such as a program, is secure by noninterference without considering how that program will be used in practice. What if we wanted to hold systems to a higher standard that takes into consideration the processes that generate a system's data? For this we need a stronger security property.

We can be more specific and introduce a security policy model that is strictly stronger than noninterference.

Definition 34 (ECS semantic security). For a given ECS model, $\mathcal{Y}$ exhibits *semantic security* iff

$$\mathcal{S}_H \perp\!\!\!\perp \mathcal{A}_L$$

Semantic security is a well known property of cryptographic systems that means, intuitively, that an attacker intending to determine the contents of a message might as well not look at an encrypted signal. The term has taken on a wider use in the differential privacy literature as it has been introduced as a desideratum for statistical databases along the lines of that proposed by Dalenius in 1977 [58, 7]. Dwork and Naor [7, 27] show that it is impossible to guarantee the semantic security of such a database given arbitrary auxiliary knowledge.

We draw on the spirit of this literature in our definition of ECS semantic security. The principle difference between ECS semantic security and noninterference is that the latter is concerned with the Independence of system outputs from sensitive inputs *conditioned* on the inputs, whereas the former takes into consideration how environmental correlations may allow system outputs to reveal system inputs.

Noninterference does not imply semantic security. Put another way, the same system can be secure by noninterference but semantically insecure. Consider the system in Figure 4.4.

This system is safe when A_H is not in the conditioning set. It is secure by noninterference because when S_L is in the conditioning set, the path from S_H to A_L that goes through E is blocked. But when S_L is not in the conditioning set, this path is open and therefore A_L can be conditionally dependent on S_H .

Our conjecture is that semantic security cannot be guaranteed by the system designer alone. We are able to prove sufficient conditions for semantic security by including a general property of the world, including the environment outside the system.

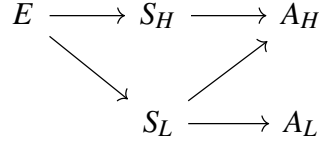


Figure 7: A system that is not propitious when E is unobserved

Definition 35 (Propitious). The world $\mathcal{W}$ is *propitious* iff there is no unblocked path between S_H and S_L .

Theorem 36. If a system is present, safe, and orderly, and the world is propitious, then the system is semantically secure.

Proof. By Theorem 33, the system is secure by noninterference, implying that

$$S_H \perp\!\!\!\perp A_L | S_L$$

So no unblocked paths run from S_H to A_L when S_L is in the conditioning set.

Recall from the proof of Theorem 33 that this was because we ruled out all possible paths from S_H to A_L .

Paths running from S_H through S_L to A_L were blocked because S_L was in the conditioning set.

Consider any such path, now unblocked as S_L is not in the conditioning set.

If it is unblocked, then there is an unblocked path between S_H and S_L , which contradicts the assumption that the world is propitious.

Therefore there are no unblocked paths between S_H and A_L and so $A_L \perp\!\!\!\perp S_H$.

□

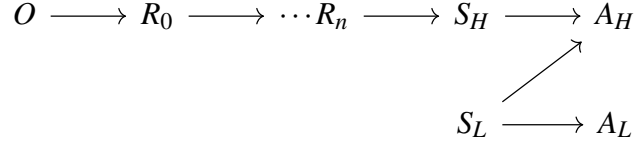
4.8 Formalizing origin privacy

We have defined origin privacy as privacy policies that place restrictions on information based on its provenance. This is in contrast to policies that restrict information based on its content. Another way to put this difference is that origin-based privacy policies restrict information based on the structure of causal flows, while information content based policies restrict information based on its nomic associations.

The problem with restricting information flows based on information content is well illustrated by the problem of guaranteeing semantic security in an ECS system. Any sensitive information content can potentially have nomic associations with otherwise innocuous inputs due to causal paths in the environment. Guaranteeing the absence of associations depends on properties of the environment that may be outside the system designer's control. Noninterference, on the other hand, is an achievable property for a system designer. However, it is defined in such a way that it can be guaranteed even when some kinds of harmful information leaks are probable in practice.

In our legal analysis in Section 4.1 we identified some policies that restrict information based on its provenance, or origin, rather than its information content. In Section 4.2, we have identified the origin of information as the chain of processes causing its input into the system. Taking the concept of “high-side” input as those inputs to a system that are treated with special sensitivity,

we can model an example world that meets the most basic requirement of an origin based policy roughly like so:



In this model, the original value O is connected only to the high-side input S_H by a direct path of relays $R_0, \dots, R_n$. We can define the origin property as:

Definition 37 (Origin restricted). A system $\mathcal{Y}$ with inputs $\mathcal{S}$ is *origin-restricted* for a protected variable O iff all direct paths from O to $\mathcal{S}$ end in $\mathcal{S}_H$, and there is at least one such path.

In what sense is an origin restricted system secure? We would like the low-side output of an origin restricted system to be independent of the sensitive variable. As we have seen in Sections 4.7.3 and 4.7.4, there are multiple security models that make different assumptions about the conditions of security. We can use analogous security models for origin privacy.

Definition 38 (Origin noninterference). $\mathcal{Y}$ is secure by origin noninterference with respect to a sensitive variable $O \in \mathcal{E}$ iff

$$A_L \perp\!\!\!\perp O | S_L$$

Theorem 39. Given an ECS model, if the system is present, safe, orderly, and origin restricted then the system is secure by origin noninterference.

Proof. Because the system is origin restricted, there is at least one direct path from O to $S_H \in \mathcal{S}_H$.

If there were an unblocked path from A_L to O that included an outgoing edge from A_L , this would extend into an unblocked path to S_H , violating the condition imposed by Lemma 32. Therefore there is no unblocked path from A_L to O that includes an outgoing edge from A_L .

Because the system is orderly, incoming edges to A_L must go to nodes in the system. Therefore, any unblocked path from O to A_L must go through $\mathcal{S}$.

Because the system is safe, there is no unblocked path from $\mathcal{S}_H$ to A_L .

Because the system is orderly, any path from $E \in \mathcal{E}$ to $Y \in \mathcal{Y}$ going through $\mathcal{S}_L$ will include a head-to-tail triplet centered on S_L . Conditioning on this node S_L blocks the path.

Therefore there is no unblocked path between O and A_L , and the system is secure by origin noninterference. $\square$

Definition 40 (Origin semantic security). $\mathcal{Y}$ is secure by origin noninterference with respect to a sensitive variable $O \in \mathcal{E}$ iff

$$A_L \perp\!\!\!\perp O$$

Theorem 41. Given an ECS model, if the system is present, safe, orderly, and origin-restricted and the world is propitious, then the system is secure by origin semantic security.

Proof. By Theorem 39, the system is secure by origin noninterference.

The system is origin restricted, implying that there is at least one direct path from O to S_H . S_L cannot be on this path because the system is orderly. As no node on this path is in the conditioning set, it is not blocked.

Mirroring the proof to 36, we consider any path ϕ between O and A_L that was blocked by conditioning on S_L . Such path must have a node S_L either within a head-to-tail triplet or as a common cause.

Suppose ϕ includes S_L in a head-to-tail triplet. Then there is a subpath of ϕ there is an unblocked path between S_L and O . But there is also an unblocked path from O to S_H , implying that there is an unblocked path from S_L to S_H . This contradicts the condition that the world is propitious.

Suppose ϕ includes a S_L as a common cause node. Because the system is orderly, both outgoing edges must go to nodes in $\mathcal{V}$. The path ϕ must therefore enter the system through a node in S_H . That implies a subpath of ϕ within the system runs unblocked from S_H to S_L . That contradicts the condition that the world is propitious.

Because no unblocked path between O and A_L is possible, $O \perp\!\!\!\perp A_L$ and the system has origin semantic security. □

This demonstrates that origin restrictions do prevent associations between low-side outputs and the sensitive environmental variable under the condition that the systems are otherwise secure.

4.9 Use case: IoT and biometric data

In this section we introduce a use case of Origin Privacy that we have identified through legal analysis and conversations with stakeholders.

Example 42 (Smart building biometric sensing). In an “Internet of Things” instrumented building, many sensors collect information about the contents of rooms, including photograph and other imagery such as infrared scanning to identify the number and size of people present. This information is useful to control the environment in the room (heating, ventilation). However, this data can also be potentially identified using auxiliary information, such as a facial recognition database. This processed data reveals the identities of persons in the room. In some cases this may be intentional, as when it is used for building security. In other cases these revelations may be unexpected and constitute an invasion of privacy.

We chose this example because it highlights the way smart building technology interacts with privacy policies around photography and biometric data.

4.9.1 GDPR biometric data

Here we focus particularly on the EU’s General Data Protection Regulation (GDPR) (Regulation (EU) 2016/679). In general, the GDPR places a number of restrictions on the processing of *personal data*, which it defines thus:

‘personal data’ means any information relating to an identified or identifiable natural person (‘data subject’); an identifiable natural person is one who can be identified,

directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person; (Article 4 §1, GDPR)

We interpret this definition as referring to the topic or content of information; personal data is any information *relating to* a natural person. As we have argued, a system designer cannot guarantee that a system does not process information relating to a natural person since these relations may be caused by nomic associations that are external to the system itself.

Noting this difficulty with ensuring compliance, we can nevertheless continue to work with the more specific requirements relating to biometric data. In particular, the GDPR makes a distinction between photographs and biometric data:

The processing of photographs should not systematically be considered to be processing of special categories of personal data as they are covered by the definition of biometric data only when processed through a specific technical means allowing the unique identification or authentication of a natural person. (Recital 51, GDPR)

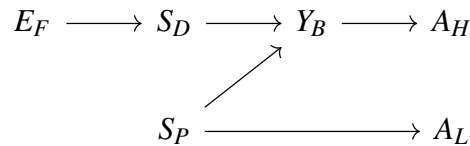
By definition under the GDPR, biometric data is a form of personal data that results from particular kinds of processes:

‘biometric data’ means personal data resulting from specific technical processing relating to the physical, physiological or behavioural characteristics of a natural person, which allow or confirm the unique identification of that natural person, such as facial images or dactyloscopic data; (Article 4 §14, GDPR)

Unlike the definition of personal data, the definition of biometric data is an origin requirement because it refers to the causal flow of data from a class of processes. Using these legal requirements, we can now use Origin Privacy to formalize their semantics with respect to a system.

4.9.2 Formalizing GDPR requirements

Consider the smart building example as an ECS. Let S_P be the photographic input to the system. Let S_D be a database of identified photographs, originating from an external process E_F . Let Y_B be a component of the system $\mathcal{Y}$ caused by S_P and S_D ; it includes imagery from S_P that has been identified using the database S_D .



We note that the photographic input S_P may indeed “relate to” natural persons in a systematic way if, for example, certain persons frequent the smart building on a regular schedule. Since these regularities (nomic associations) are unknown to the system designer, there is little she can do to guarantee the treatment of this information as personal data. What the system designer does have access to is the identified faces database, S_D . The *process of identification* that results in the biometric building surveillance data Y_B requires data from an identified source such as S_D .

The system designer knows about the origin of S_D . Specifically, she knows that this data is sourced from E_F , a process that personally identifies the facial images within it. Knowing the environmental source is sensitive, they can impose the conditions for noninterference between E_F and A_L : that no unblocked path exist within Y between inputs originating in E_F and A_L . This implies that both S_D and Y_B be d-separated from A_L within $\mathcal{Y}$. Note that in the diagram above, E_F is indeed d-separated from A_L when the system is *covered*, i.e. when none of its components are in the conditioning set. Intuitively, Y_B is subject to restricted flow because it originates from the sensitive process E_F ; it inherits this origin from one of its parent components, S_D .

We build on this result to model more complicated aspects of GDPR compliance. For example, processing of personal information, including biometric information, is generally legal given the consent of the identified person. We can introduce identified sources into the ECS model by denoting the set of natural persons $\mathcal{I}$, and denoting a process that generates data from an identified person X_i for $i \in I$. We can then place conditions on any data that is a result of causal flow from this source, as in this example specification:

Example 43 (Disclosure specification). In the system, all outputs $A \in \mathcal{A}_L$ such that A is a descendant of X_i must also be a descendant of Z_i , where $i \in I$ is the identifier of a natural person, X_i is personally identifiable information, and Z_i is a disclosure agreement identifiable with that person.

To the extent the GDPR controls on biometric information are use restriction or topic restrictions as opposed to an origin restriction, they cannot be automatically enforced based on origin alone. However, considering GDPR through the rigor of Origin Privacy clarifies some of its requirements as formal specification shows what knowledge is needed by the system designer for origin based policy enforcement.

4.10 How to Author Policies for Origin Privacy

4.10.1 Overview

In this section, we now turn to the question of how to select an Origin Privacy policy. Origin Privacy provides a way of looking at the enforcement problem, one that does not capture all of privacy enforcement but does provide a useful and automatically enforceable subset of it. Similarly, we consider an incomplete but useful way of looking policy selection.

Origin Privacy focuses on the consequences of either sharing or not sharing information from each origin. It is a consequentialistic take on privacy, asking not whether some state of affairs violates a pre-ordained privacy rule, but rather whether some sharing of data would cause harm and how a policy can mitigate this risk. A well crafted Origin Privacy policy will promote the most valuable data sharing by suppressing less valuable ones.

For example, consider a patient discussing health concerns with a physician. The patient faces a choice as to whether to mention his drug use, and the physician faces a choice about whether to

Table 1: Outcomes from sharing and using information

	Patient tells physician	Patient does not tell physician
Physician would tell police	Patient well treated but arrested	Patient not well treated
Physician would not tell police	Patient well treated	Patient not well treated

report it to the police if told. These choices lead to outcomes of varying qualities for the patient. (See Table 1.) If the patient would rather receive poor treatment than being arrested, he might decide to not tell the physician since it is the only way to guarantee not being arrested. If the physician would like to treat patients well, the physician should take telling the police off the table, for example, by adopting a public policy of not telling the police. In this case, the patient would have nothing to lose by telling the physician, and healthcare would be better served.

Complicating matters, healthcare is not the only concern. The case in which the patient is arrested is the only case in which drug laws are enforced. Society might value the enforcement of its laws for a wide range reasons, including ensuring that drug addicts receive treatment with or without their consent. This tension might appear to be an argument for the physician telling the police since it might even improve healthcare. However, note that, threatened with arrest, most patients will simply not tell their physicians about their drug use, no arrests would be made, and healthcare will not improve. The tension proves illusionary once one considers how patients would react, that is, how those controlling the origin of the data will react.

Such reasoning would not apply to mandatory and unavoidable drug tests. In this case, people (not just those who opt to become forthcoming patients) have no choice but to share information and this required sharing of information has no obvious effects upon the sharing of other information. However, this does not mean that such testing is privacy respecting: again, Origin Privacy only provides one way of looking at privacy policies. A policy requiring drug tests may violate privacy in other ways. Indeed, privacy is both a multifaceted concept [26] and an essentially contested concept [59].

Rather than attempt to solve all of privacy, focus on just the origin-view of policy selection. We first make some preliminary remarks about modeling contexts and then present the patient–physician example in more detail, using it to motivate and explain our approach to selecting Origin Privacy policies. We will end with some more general remarks about policy selection and the limitations of our approach.

4.10.2 Modeling Contexts

Origin Privacy, being an adaptation of Contextual Integrity [4], conceives of interactions between people and flows of information as happening within some *context*, a social sphere consisting of actors operating in some roles to further the ends of the context. For example, the medical context includes people acting in the role of patients and people acting in the role of healthcare providers. They, along with others acting in other roles, attempt to ensure the health of the patients, the goal of the context.

Those acting in the roles can expect certain behaviors from others acting in other roles. These behaviors represent the *norms* of the context. Some norms are clearly inherent in the nature of the

role, for example, healthcare providers provide healthcare. Others are less obvious, but nonetheless present. As an example, the norms surrounding privacy in healthcare are both well established and not inherent to the provision of healthcare. (Indeed, they are largely absent for patients who are prisoners.) Rather, they have arose over time as it was discovered that healthcare providers who did not share what they heard from patients had more open relationships with their patients, got more complete information about them, and were able to provide higher quality care. Eventually, these informal norms got codified into law as part of the Health Insurance Portability and Accountability Act of 1996 (HIPAA).

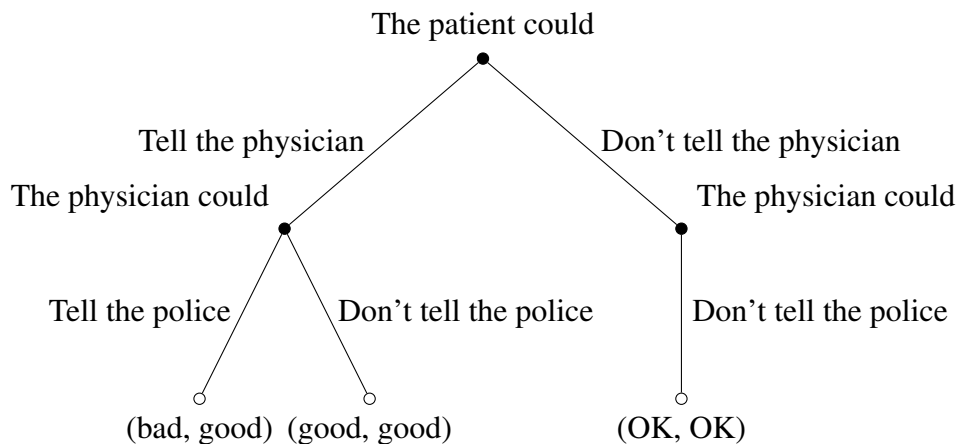
This process illustrates that norms can have complicated relationships with the goals of the context that they govern. To select a policy, such as HIPAA, the policymaker must understand the context and how the policy will affect the behavior of its members. A systematic way of reasoning about such behaviors is game theory. While a full game theoretic analysis of a context will often be too expensive to conduct and game theory does not capture humans' irrational tendencies, we will presume that such an approach is followed for policy selection.

The exact game model appropriate to a context will depend upon the nature of the choices that people face about the actions to take in that context. No matter the exact model, we map the roles of the context to the players of the game since the individual identities of people in the roles typically do not matter. These players face choices about how interact with one another. From a privacy perspective, of particular interest are choices about the sharing of information. The sharing of information can enable a player to make more informed choices, meaning that one choice can affect the ability to make another choice. Depending upon whether their incentives are aligned, one player might or might not view another player's improved ability to select actions as an improvement. For example, a drug-using patient would view a physician's ability to make better choices about healthcare as good but maybe not a police officer's ability to make better choices about who to arrest. We can model these issues using games with *imperfect information*.

4.10.3 Patient–Physician Example

The physical constraints of reality provides a description of how players could act in the context without any norms. The matrix shown in Section 4.10.1 provides such a description. As with all models of reality, it is an abstraction. We assumed that the police will always make an arrest and, therefore, do not model their choices. We also focused only on the choices related to the sharing and did not model why the physician knowing about the patient's drug use will enable the provision of better healthcare. Despite these and other abstractions, this model is sufficient to illustrate some points about policy selection.

The *extensive form* of the game makes the sequential nature of the players moves explicit:



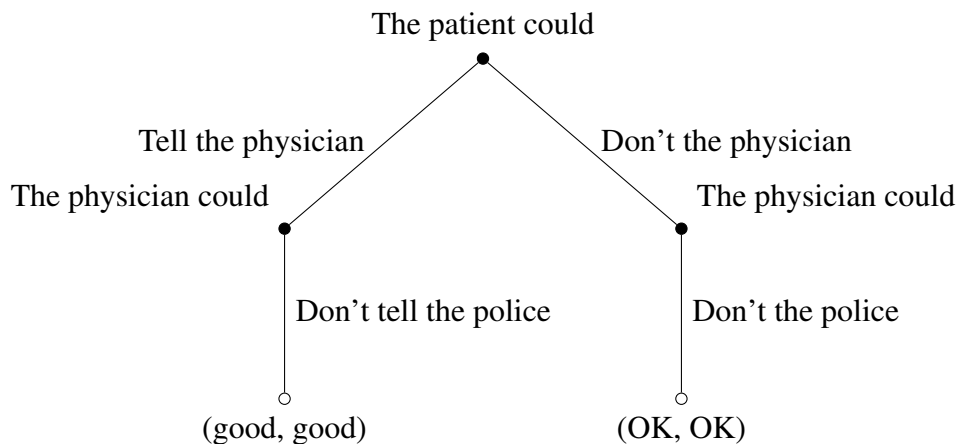
The diagram shows the starting state at the top. It then shows a choice made by the patient that leads to one of two states. From each state, the physician makes a choice leading to a ending state. The diagram shows the payoffs to the patient and physician for each ending state as an ordered pair (patient's payoff, physician's payoff).

For the patient, the payoff could represent the quality of the healthcare, since his goal is to gain healthcare. However, this would ignore the large negative utility that the patient would experience if arrested. Thus, we include this in the patient's payoff. Note that this illustrates the need to carefully consider what to include in the model. Had we not modeled the police arresting the patient, then the consequences and privacy concerns of the physician telling the police would be missing and we might had concluded that the physician telling the police is a non-issue.

For the physician, the payoff also represents the quality of the healthcare since the goal of the physician role in the healthcare context is providing high-quality healthcare.

If the patient selects to not share the information, he is guaranteed a medium degree of healthcare. If the patient selects to share the information, the physician might from the resulting state select an action that results in the lowest possible payoff or might select an action that results in the highest possible payoff. Thus, a risk-adverse patient would select to not share the information, resulting in a medium payoff.

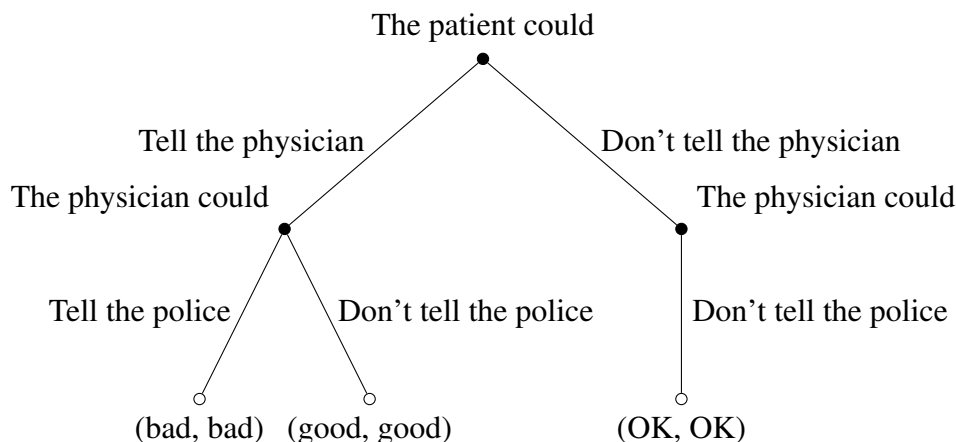
This outcome is sub-optimal for both the patient, the physician, and the healthcare context as a whole. Thus, the healthcare context should create a norm that the physician never selects to share the information with the police. This norm restricts the set of possible actions available to the physician:



Policy enforcement takes us from the first game diagram to the second. That is, it eliminates some physically possible but undesirable possible actions.

For physicians to simply say that they will not tell the police is similar to a *non-credible threat* in game theory, that is, a threat to punish even if doing so harms one's utility. Non-credible threats do not, in theory, play any role in the selection of actions since those so threatened will not take the threat seriously. Similarly, a patient merely told not to worry about having his drug use reported to the police may ignore being so told and continue to withhold such information. To make the promise credible it must be combined with an enforcement mechanism ensuring that physicians do not report drug use to the police.

The simplest such enforcement mechanism would be a publicly posted policy constituting a court-enforceable contract between the physician and the patient. In this case, the patient could sue the physician for damages, which would both discourage the physician from reporting and provide the patient with restitution in the case of a policy violation. Since the patient would receive a payout for successfully suing the physician, this enforcement mechanism is a credible threat. In essence, it coverts the first game diagram to the following one:



Since telling the police now results in a bad outcome for the physician, the physician will select not to do so, leading to a good outcome for both.

In practice, the contract is typically not between the patient and the physician but rather between the patient and a corporation, such a hospital or physicians' group that employs the physician. In this case, the corporation has a vested interest in ensuring that its employees obey the

policy to avoid being sued. This can lead to the corporation adopting automated enforcement mechanisms designed to make it difficult to violate the policy or, at least, easy to discipline those who do so. Such mechanism cannot only prevent law suites, but also give patients more trust that the policy will be obeyed, making them more willing to share information.

4.10.4 Steps for Policy Selection and Enforcement

We can generalize from the example presented above. The basic steps are as follows. First, the policymaker must figure out what payoffs or utility functions it wishes to optimize. Often this can be found from the purpose of the context in which the policymaker operates.

Second, the policymaker must model the context to understand which actions lead to optimizing the utilities. The policymaker must also model the incentives of the roles of the context to determine how they diverge from the utility function to be optimized. Such modeling can be done using game theory.

Third, the game model must be examined to determine which actions can lead to poor outcomes despite being attractive to actors since they can increase that actor's payoff. These actions should be ruled out by a policy.

Fourth, the policy must be enforced. The enforcement can take a combination of making actions impossible or altering payoffs so that the users tempted to take sub-optimal actions will no longer find the action tempting.

As far as Origin Privacy is concerned, special attention must be paid to those actions corresponding to the sharing of the information and how those actions affect the abilities of actors in the context to optimize the utility.

Actions providing information can either lead to an increase or decrease in utility. Those that only increase utility should be encouraged. Those that only decrease utility should be discouraged or disabled. More complicated are those that can either increase or decrease utility depending upon the other actions. In that case, a policy can discourage the other actions that decrease the utility while encouraging those that increase it. If successful, the information-providing action can be encouraged. This is a self-serving motivation for information use control.

Two major difficulties exist in following these steps. The first is selecting the utility function to optimize. Reasonable people can disagree about the exact purpose of a context or even what the context is. The second is producing a model of the context accurate enough to predict how actions will affect the utility.

In practice, resolving these problems perfectly is often not required. Despite disagreements over the details, in some cases, the effects of providing information will either be positive or negative under a variety of utility functions under a variety of models. Agreement about whether it is positive or negative can be enough to motivate a policy.

Finally, following these steps ignores that contexts do not exist in a vacuum. A single action can exist in multiple contexts with varying purposes, as illustrated by telling the police affecting both healthcare and law enforcement. Resolving these cross-context conflicts is currently beyond the theory of contextual integrity.

5.0 CONCLUSIONS

Origin-based information-flow rules can be seen as an extension of noninterference and semantic security, two well-known security properties. Origin restrictions extend the security properties of a system towards a sensitive origin event. We are able to capture other notable data-protection concepts in this framework, including the GDPR's rules on biometric data and a variation of differentially private databases. In many cases, origin-based rules can replace subject- and topic-based rules, leading to a more clear and more enforceable policy. Origin-based policies can be specified in a Legalese-like language and enforced using a label-tracking system.

We have analyzed privacy policies and discovered that they variously restrict information based on origin and topic. We developed an informational ontology that reflects the assumptions underlying many of these policies. We have shown that this informal ontology can be formalized in terms of causal graphical models. These models show that the two aspects of information flow correspond to precisely defined concepts of causal flow and nomic association. Both senses of information flow are accommodated by an understanding of situated information flow as a causal flow in causal context, as represented by a Bayesian network.

We developed a model of system security, the ECS model, which represents a system embedded in its environment. This model can demonstrate and extend known results in computer security, such as those concerning noninterference, semantic security, and differential privacy. It also allows us to formally define a new privacy property, origin privacy, which assumes that system designers have some control over the paths through which information enters their systems. We demonstrate how the ECS model can be used to elucidate a case of implementing GDPR compliance on biometric data. We demonstrated preliminary results on how the ECS model can be extended into game theoretic form to account for how strategically acting agents interact with systems with or without relevant security properties.

These contributions are suggestive of several lines of future work.

5.1 Specifiability Criterion

One direction for future work is to question what these results mean for the design and interpretation of legal policies. Are privacy policies based on information topic harder for consumers to understand than policies based on information origin? How do end users interpret ambiguous language in privacy policies?

One benefit of using causal models as opposed to program analysis for considering information flow security of technical systems is that it shows that information flow security is not only a technical problem. Because information leaks based on nomic associations may be properties of any mathematically defined causal system, these results extend to human institutions as well. In general, mathematical results about the limits of computational enforceability will generalize to the possibility of enforceability through non-computational means.⁶ Privacy policies and social expectations that restrict information based on its contents may be unenforceable or ambiguous in general.

Concretely, the brittleness of these kinds of restrictions is exposed by advances in large-scale data analysis, or "big data", which routinely confound our expectations about what information is

⁶This claim assumes the Church-Turing thesis.

about. Data about a person's purchases of scent-free hand lotion, which we might have assumed to be innocuous, has been shown to be correlated with sensitive information about early-stage pregnancy [2]. Ad targeting systems that use all available correlational information in the data they collect risk violating people's privacy due to unexpected discoveries from automated learning processes.

By demonstrating the limits of what security properties can be enforced, and by whom, this framework can shed light on how privacy policies should be written and which parties should be held liable in the event of a violation. This may build on other work in identifying legal liability in cases of perceived inappropriate behavior of a sociotechnical system [60].

5.2 Observer capabilities

We have shown that nomic associations between system outputs and sensitive environmental variables can lead to violations of privacy policies. In order for these threats to be material, nomic associations must be known to attackers as auxiliary information. A natural next step in this line of inquiry is a more systematic study of how observer's capabilities for learning nomic associations factor into information flow security considerations.

In our models in this article, we have used Bayesian networks as models of the objective frequencies of variable outcomes. Possible inferences from observed variables have been interpreted as those inferences possible *in fact* from the causal structure of the world. Information flow security guarantees were possible when the system designer was assumed to have some true knowledge about the system's environment, such as the origin of its inputs.

In practice, most systems will be embedded in environments that are only partially known. In the cases of fraud and spam detection, and other adaptive machine learning systems, a model of the origin of inputs is trained continuously from collected data. Probabilistic graphical models are indeed one of the many machine learning paradigms used in these applications [61].

A direction for future work is developing a theory of information flow security and privacy under conditions where observer knowledge of nomic associations is itself a function of system inputs. In simple cases this may reduce to single program analyses already established in work on use privacy [19]. In cases where multiple systems interact, there may be novel problems.

5.3 Incentives-based policy design

We have provided an informal sketch of how to select origin-based privacy policies by examining the incentives of the parties involved (Section 4.10). While we believe this to be a reasonable way of selecting policies, more rigor can be brought to its practice.

Prior work has been done on game theoretic models of information flow policies [13], mechanism design with differential privacy [62], and semantic interpretations of differential privacy framed in terms of the incentives of data subjects [63]. We see potential to continue this line of work using statistical models of systems, their internal processes, and their environment, including the process that generate their input data.

We have shown how Multi-Agent Influence Diagrams (MAIDS) [24] can be used in game theoretic modeling of security problems. Feng et al. [55] have used Bayesian networks to model security risks and perform a security vulnerability propagation analysis. They build their causal model from observed cases and domain experts. We anticipate new frameworks for aligning the

incentives of system designers and data subjects which are sensitive to risk of data misuse (see [9]). An application of this work is automated policy design for smart buildings and cities, where interacting data subjects and control systems must share information while minimizing data misuse.

Our work in Section 6.4 is a first step in developing a new way of assessing the value of security properties based on their impact on game outcomes. This method of modeling information value through data games is the subject of the next chapter.

We have unpacked the assumptions of privacy policies to develop a general ontology of systems, processes, and information. We have then formalized this ontology using the assumption that these systems are subject to the laws of probability and an interventionist account of causation [43]. Using these simple assumptions, we have confirmed well-known results about information flow security about programs and systems in isolation. We have also gone beyond these results by showing explicitly what their consequences are when systems are embedded in an environment, developing a general security modeling framework for this purpose.

We prove the relative effectiveness of origin based information flow restrictions over association based information flow restrictions using the causally embedded system framework. We show how origin privacy would be applied in a GDPR and Internet of Things use case. We anticipate new lines of inquiry extending from the intersection of causal modeling and information flow security, including evaluation of policy enforceability, modeling the role of observer knowledge in privacy and security, and automated policy generation through incentives-based design.

6.0 RECOMMENDATIONS

6.1 Declassification

Section 4.1.1 mentions using syntactic interpretations of information topics for expressing exceptions to more general rules. Such exceptions could be seen as a method of declassification. The question of when it is safe to release some data warrants further investigation as does how to structure such exceptions.

6.2 Purpose

In addition to finding the concept of data origin in privacy policies, we found references to the purposes for which data may be used. Unlike restrictions based on the subject and topic of data, we typically did not find it possible to reduce such purpose-based restrictions to the data's origin. However, much like the difficulty of pinning down data's subject and topic, pinning down the purpose behind data use can be difficult, requiring information about the user of the data that is often unavailable to the policy enforcer [11, 18]. How to combine origin-based restrictions with purpose-based ones warrants further exploration.

6.3 Relationship to Differential Privacy

While differential privacy is built on firm theoretical foundations, sometimes its application to a particular problem is not. Furthermore, researchers often use weakened versions of differential

privacy. We recommend further research on the practice of differential privacy and variants to understand how to apply them in a privacy-preserving way. This will require relating these definitions to privacy conceptions that are closer to intuition, such as Contextual Integrity.

Towards this, this section will show a connection between origin privacy and differential privacy. Thus far we have defined origin privacy strictly in terms of conditional independence. This has been inspired by the formal security model of *noninterference*.

When assessing computational systems that process personal information, it is possible for privacy requirements to be looser than this strict security requirement. The particular case of privacy preserving statistical analysis of databases of personal information has motivated differential privacy as a formal privacy model [7, 27].

Formally, an algorithm $\mathcal{A}$ is ϵ -differentially private if for all subsets $S \subset \text{image}(\mathcal{A})$ and datasets D_1 and D_2 that differ on a single element, the probability of the output of the algorithm run on the datasets being in S differs by at most a multiplicative factor of e^ϵ and an additive factor δ [64].

Definition 44 (Differential Privacy ((ϵ, δ)-DP)).

$$\Pr[\mathcal{A}(D_1) \in S] \leq e^\epsilon \Pr[\mathcal{A}(D_2) \in S] + \delta$$

When $\delta = 0$, then $\mathcal{A}$ is ϵ -differentially private (ϵ -DP).

Consistent with our origin privacy approach, we will investigate how differential privacy can be assessed given a model of system and its environment as a causal Bayes network. We will draw on prior results relating differential privacy and causality and variations on differential privacy expressed in terms of mutual information.

6.3.1 Mutual information differential privacy

Cuff and Yu [65] demonstrate that there is an equivalence between differential privacy and what they define as mutual information differential privacy:

Definition 45 (Mutual information differential privacy (ϵ -MI-DP)). A randomized mechanism $P_{Y|X^n}$ satisfies ϵ -mutual information differential privacy if,

$$\sup_{i, P_{X^n}} I(X_i, Y | X^{-i}) \leq \epsilon$$

where X^{-i} is the dataset X excluding variable X_i .

They prove that ϵ -MI-DP is equivalent to differential privacy in the sense that ϵ -MI-DP implies (ϵ, δ)-DP for some δ , though ϵ -MI-DP is weaker than ϵ -DP. McSherry [66] argues that that ϵ -MI-DP falls short of the desiderata of ϵ -DP. Nevertheless, we will proceed with the ϵ -MI-DP because it is suggestive of probabilistic structure may be used to infer a privacy relevant bound.

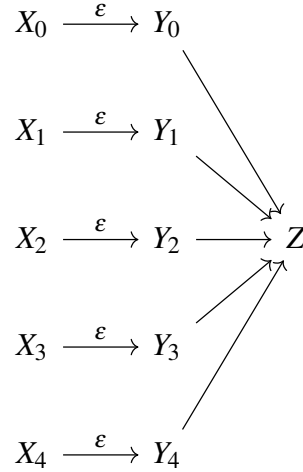
That the mutual information limit is conditioned on every other member of the database is an indication of a disappointing fact about differential privacy, which is that its beneficial properties with respect to preserving privacy are not robust to cases when entries in the database are correlated with each other.

The value of using MI-DP for our purposes is that the properties of mutual information are well-understood, and we can derive a number of useful theorems about mutual information between variables in a Bayesian network.

6.3.2 Randomizing database inputs

We can now combine the previous results to show how a system designer can develop a Bayesian network model that guarantees differential privacy. A common way of achieving differential privacy is by randomizing inputs to the database prior to aggregation; this is done in practice in Erlingsson et al. [67]. We can model randomization explicitly as in the following example.

Let X_i be a set of variables representing the personal information to be aggregated. Let Y_i be a random variable over the same domain as X_i that is almost but not quite independent of X_i ; we'll say that the mutual information between X_i and Y_i is bounded by ϵ_i . Then aggregate all the Y_i variables into a database, Z , that is available for querying.



In the above diagram, we annotate an arrow between variables A and B with the upper bound on the mutual information $I(A, B)$ afforded by the conditional probability distribution. In this case, we have set all the ϵ_i equal to each other, ϵ .

We can now use this graphical structure to prove that this system is 2ϵ -MI-DP). We can prove this using the Data Processing Inequality,

Proposition 46 (Data Processing Inequality). If three variables are in a Markov chain

$$X \rightarrow Y \rightarrow Z$$

where $X \perp\!\!\!\perp Z|Y$, then $I(X, Y) \geq I(X, Z)$

A standard proof of this is in the appendix. We will also use the Path Mutual Information Theorem (PMIT), that we prove as Theorem 65 in Appendix A.

Example 47. For the structure described above, Z is 2ϵ -MI-DP.

Proof. Note that because X_i and X^{-i} are joined only by a path with a common effect node, $I(X_i, X^{-i}) = 0$.

It follows that:

$$\begin{aligned}
& I(X_i, Z | X^{-i}) \\
&= I(X_i; Z, X^{-i}) - I(X_i, X^{-i}) \\
&= I(X_i; Z, X^{-i}) \\
&= I(X_i; Z) + I(X_i; X^{-1} | Z)
\end{aligned} \tag{9}$$

By DPI and the graphical structure, we know that for all i

$$I(X_i, Z) \leq I(X_i, Y_i) = \varepsilon$$

By PMIT, we know the mutual information of two variables connected by a path with all of its common effect nodes observed is bounded by the mutual information of steps along the path. In this case, it entails that:

$$I(X_i; X^{-1} | Z) \leq I(X_i, Y_i) = \varepsilon$$

By substitution, we know that:

$$I(X_i, Z | X^{-i}) = I(X_i; Z) + I(X_i; X^{-1} | Z) \leq 2\varepsilon$$

As this holds for all i , it follows that Z is 2ε -MI-DP. □

6.3.3 Generalizing ε -security

We can generalize the two formal security models that we introduced in Section 4.7.3 to models that allow for ε mutual information between sensitive inputs and low-side outputs.

Definition 48 (ε -noninterference). A system is secure by ε -noninterference iff

$$I(A_L, S_H | S_L) \leq \varepsilon$$

.

Definition 49 (ε -semantic security). A system is ε -semantically secure iff

$$I(A_L, S_H] \leq \varepsilon$$

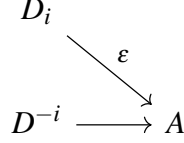
.

Recall that two variables are perfectly independent if and only if their mutual information is zero, implying that 0-noninterference is equivalent to noninterference, and 0-semantic security is equivalent to ECS semantic security.

We can now show that differential privacy follows from an application of ε -noninterference with one important caveat. We have defined noninterference in terms of a system's high-side and low-side inputs. Schematically, we have considered the "high side" to be a part of the system in need of special information flow restrictions, while the "low side" is what's available to less restricted access.

This model is intended to be generalized to cases where there are multiple categories of restricted information. In particular, to prove that ε -noninterference implies differential privacy, we must consider *each entry individually* to be noninterferent with respect to the other entries in the data set.

Theorem 50. For an ECS model, if for all D_i , ECS is secure by ϵ -noninterference with respect to D_i as a high-side input and D^{-i} as low-side input, then A is mutual information differentially private with respect to data set D



Proof. If for all D_i , ECS is secure by ϵ -noninterference with respect to D_i as a high-side input and D^{-i} as low-side input, then

$$\forall i, I(D_i, A_L | D^{-i}) \leq \epsilon$$

which implies that

$$\sup_{i, P_{X^n}} I(D_i, A_L | D^{-i}) \leq \epsilon$$

which is the condition for ϵ -MI-DP. □

It is not generally the case that if each of a database's entries D_i is ϵ -semantically secure from the output A that the output will be ϵ -differentially private. A bound on $I(D_i; A)$ does not imply a bound on $I(D_i; A | D^{-i})$, as is apparent from Equation 10.

$$I(D_i; A | D^{-i}) = I(D_i, D^{-1}; A) - I(D_i; A) \quad (10)$$

6.4 Incentives in ECS

We consider the selection of privacy policies in Section 4.10. However, our model of policy selection only looks at one way of doing so and is far from precise. As discussed in our note on gaps in privacy presentations [25] (and included as Appendix C), research teams often give little attention to this pressing and complex problem. We recommend further research into how to select privacy policies, particularly with a focus on balancing trade-offs between concerns. Here we consider an approach such research could take.

A discovery during our analysis of origin privacy and differential privacy is that while many different formal security and privacy properties have been proposed, there is little formal work establishing the relationship between these properties and the incentives of those using and operating data processing systems. How do these properties align with the incentives of the actors involved with the system?

We recommend more research into this question, building on the security modeling results presented in Section 4.7.1. This section includes preliminary work along these lines. This work uses Multi Agent Influence Diagrams, a game theoretic expansion of Bayesian networks developed by Koller and Milch [24]. We prove a relationship between the strong security property of semantic security and the incentives-based property of tactical independence.

We see this preliminary result as a first step in a longer line of research that validates the intuitive claims made about the effect of system privacy on social and market outcomes. We are particularly interested in whether secure systems can guarantee not just tactical but also strategic independence of competing actors. The two notions differ in that only strategic independence prevents the chilling effects that come from surveillance.

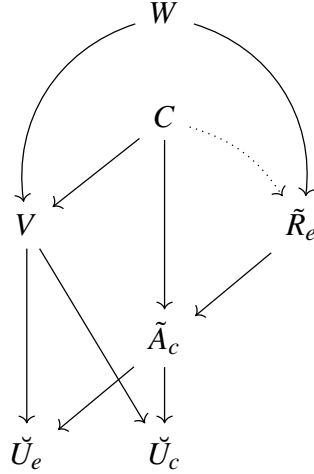
MAIDs To accomplish this, we will use the Multi-Agent Influence Diagram (MAID) framework developed by Koller and Milch [24]. In brief, a MAID is a Bayesian network with two important extensions.

- Some of the variables are reserved as *decision variables* and assigned to one of several agents. An agent's assignment of CPDs to its decision variables is that agent's *strategy*; replacing each decision variable with the CPD from a *strategy profile* transforms the MAID into a Bayesian network.
- Some of the variables are reserved as *utility variables*. These are assigned to one agent each, and are summed when realized into the agent's total utility. Utility variables must not have children.

A formal definition of MAIDS, strategies, and other useful properties is given in Appendix B, which includes an account of the graphical notation we will use in this Section.

6.4.1 Expert services model

As an illustration of an information game that can be represented as a MAID, consider the following diagram, which is a generalized model of an expert service. (This model will be analyzed in more detail in the following section, in Section 6.4.2.) The services, which include health services, legal services, as well as some software based services like search engines, involve a client, c , who presents knowledge of their situation to an expert, e . The expert has access to general knowledge relevant to the client's condition, and recommends some action based on this knowledge. The client can choose to take this recommendation. In the idealized case considered here, the client and the expert have perfectly aligned incentives and so the expert will use their knowledge to the best of their ability.



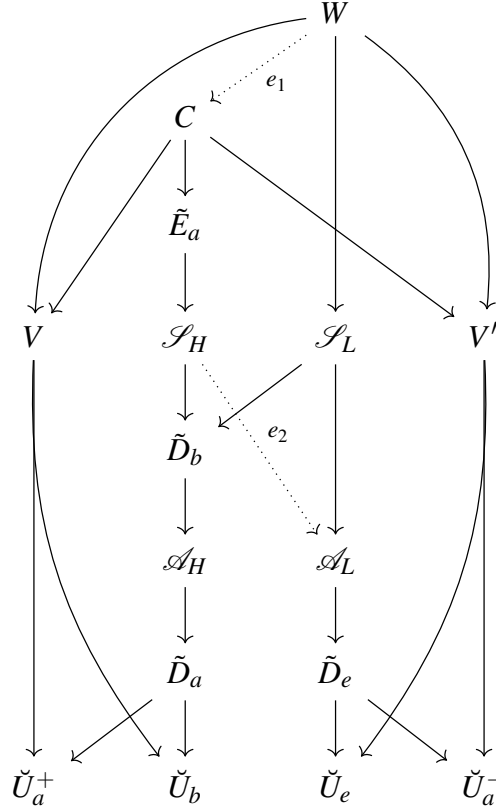
In this model, W is the generalized knowledge of the world that is available to e at their decision variable representing their recommendation, $\tilde{R}_e$. The action taken by the client $\tilde{A}_c$ and the action value function V determine the utility of both the expert and the client, $\check{U}_e$ and $\check{U}_c$. The value function is influence both by general facts about the world W and the particular situation of the client, C . The client knows their situation but not the general expert knowledge.

The client's action is informed by the general knowledge only through the recommendation of the expert. The expert may or may not know the client's specific situation; this is represented by the dotted arrow between C and $\tilde{R}_e$.

The value of such a model is that a qualitative analysis can readily provide insights into the influence of an information flow on outcomes. Since we know the expert's utility depends on influencing the client to make the best possible action at $\tilde{A}_c$, and that the value of this action depends on V , the expert's effectiveness will be limited by how well they can predict V given the knowledge available to them at $\tilde{R}_e$. Without information about their client's specific situation C , their advice can at best be perfectly general. But with access to information C , the expert can improve their recommendation and outcomes for both players.

6.4.2 Expert ECS Model

We now combine the expert service model with the ECS model. We will embed the expert insider an ECS and give them access to personal information of the client via the high-side input. They will also have access to general knowledge through the low-side input. An adversary will have access to the low-side output, but not the high-side output. Using this model, we will be able to test how the security properties we have analyzed in Section 4.7 and Section 4.8 can be motivated in terms of the way they affect the incentives of interacting with the system.



This game has three players, Alice (a), Bob (b), and Eve (e).

Alice and Bob have perfectly aligned incentives, and Eve is an attacker who is adversarial to Alice. We specify that the following relations hold:

$$\begin{aligned}\check{U}_a^+ &= \check{U}_b \\ \check{U}_a^- &= -\check{U}_e \\ \check{U}_a &= \check{U}_b - \check{U}_e\end{aligned}$$

At the center of this model is an ECS, with high- and low- side sensors ($\mathcal{S}_H, \mathcal{S}_L$) and actuators ($\mathcal{A}_H, \mathcal{A}_L$).

In this model, Alice is aware of her personal information C and decides at $\tilde{E}_a$ what if any of it to divulge secretly (via $\mathcal{S}_H$) into the system because she wants an expert recommendation from Bob. Bob has access to general expertise W through a low-side input ($\mathcal{S}_L$). These inputs are both available to Bob at his decision node $\tilde{D}_b$, at which he chooses a recommended action for Alice, which he passes through the high-side output $\mathcal{A}_H$.

Alice will use the information about the recommendation taken from the high-side output $\mathcal{A}_H$ to choose an action. The utility of this action will depend on the action values V , which are a function of two variables: the personal characteristics C of Alice and other general information about the world, W .

Eve will make a decision $\tilde{D}_e$ based on the low-side output of the system, $\mathcal{A}_L$. Eve's utility U_e depends on this decision and an action value function V' that is analogous to Alice's action value function V , in that it depends on C and W .

The system in this diagram is $\mathcal{Y} = \{\mathcal{S}_H, \mathcal{S}_L, \tilde{D}_b, \mathcal{A}_H, \mathcal{A}_L\}$.

The diagram has two dotted edges, e_1 and e_2 . Each dotted edge may be either included in the graph (open) or excluded from the graph (closed). The diagram therefore describes four distinct models: none open, e_1 open, e_2 open, and both open. We will analyze each case in Section 6.4.2.

Analysis First, we can analyze the expert ECS model presented in Section 6.4.2 in terms of the system design properties introduced in Section 4.7.2.

Note that no descendent of $\mathcal{A}$ is in $\mathcal{S}$. Therefore, the expert ECS model is present if none of $\tilde{D}_a, \tilde{D}_e, \tilde{U}_a^+, \tilde{U}_b, \tilde{U}_e, \tilde{U}_a^-$ are in the conditioning set $\mathcal{C}$.

The system is covered if none of $\mathcal{S}_H, \mathcal{S}_L, \tilde{D}_b, \mathcal{A}_H, \mathcal{A}_L$ are in the conditioning set $\mathcal{C}$.

It is plain from observation that the system is orderly. It is also clear that the system is origin-restricted with respect to the personal characteristics C : there is one direct path from C to $\mathcal{S}$ and it goes to $\mathcal{S}_H$.

We are left with several candidates for the conditioning set: $W, C, V, V', \tilde{E}_a$. Recall that the world is propitious if there are no unblocked paths from $\mathcal{S}_H$ to $\mathcal{S}_L$. There are two ways an unblocked path can happen under the conditions discussed so far. One is that either V or V' is in the conditioning set. Another is that the edge e_1 is open.

It is clear that the system is safe if edge e_2 is closed and not safe if edge e_2 is open.

Suppose that there are no variables in the conditioning set. Then by the reasoning above, the following properties of the expert ECS system hold:

- If e_1 and e_2 are closed, then the system is origin secure with respect to C both semantically and by noninterference.
- If e_1 is open, then the system will be origin secure with respect to C by noninterference, but may not be semantically secure.
- If e_2 is open, then the system may not be origin secure with respect to C either by noninterference nor semantically.

Though we have been able to show that these security properties hold on the expert ECS model, this model also reveals how these security properties do not provide all desirable guarantees a system might provide in terms of the incentives of Alice and Bob.

What can be shown is that given that the expert ECS system is semantically secure, it is also the case that $\tilde{E}_a$ and $\tilde{D}_e$ are tactically independent (see Definition 74), meaning that for any strategy specifying decision rules to each decision variable, in the induced probability distribution $\tilde{E}_a$ and $\tilde{D}_e$ are independent. In other words, at the level of tactics, Alice's choice to reveal her information to the ECS will not depend on Eve's choice of how to use the system's low-side outputs adversarially.

However, despite these security properties, we can show with this model that $\tilde{E}_a$ may *strategically rely* on $\tilde{D}_e$ (see Definition 71). This means that Alice's choice of decision rule at $\tilde{E}_a$ can depend on Eve's choice of decision rule at $\tilde{D}_e$. This can be a problem for system designers if their goal is to guarantee that the presence of Eve has no deterring effect on Alice's choice to reveal her data to the ECS.

Further exploration of the relationship between system security properties and incentives of players in this causal formalism is left to future work.

7.0 REFERENCES

- [1] Eric Barker. Credit card companies are constantly digging for weird signs you'll stop paying. *Busniess Insider*, 2012.
- [2] K. Hill. How target figured out a teen girl was pregnant before her father did. *Forbes*, 2012.
- [3] Gregory Piatetsky. Did target really predict a teen's pregnancy? the inside story. *KDnuggets*, 2014.
- [4] Helen Nissenbaum. *Privacy in Context: Technology, Policy, and the Integrity of Social Life*. Stanford Law Books. Stanford University Press, 2009.
- [5] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum. Privacy and Contextual Integrity: Framework and applications. In *IEEE Symposium on Security and Privacy*, 2006. doi: 10.1109/SP.2006.32.
- [6] Joseph A. Goguen and Jose Meseguer. Security policies and security models. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 11–20, 1982.
- [7] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10–14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006. ISBN 3-540-35907-9. doi: 10.1007/11787006_1.
- [8] James Cheney. A formal framework for provenance security. In *Computer Security Foundations Symposium (CSF)*, pages 281–293. IEEE, 2011.
- [9] Privacy risk management for federal information systems. NISTIR 8062 (Draft), 2015. NIST.
- [10] Orin S. Kerr. The mosaic theory of the fourth amendment. *Michigan Law Review*, 110, 2012.
- [11] Michael Carl Tschantz, Anupam Datta, and Jeannette M. Wing. Formalizing and enforcing purpose restrictions in privacy policies. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, pages 176–190, 2012.
- [12] Michael Carl Tschantz, Anupam Datta, and Jeannette M. Wing. Purpose restrictions on information use. Technical Report CMU-CyLab-13-005 and CMU-CS-13-116, Carnegie Mellon University, 2013.
- [13] Adam Barth, John Mitchell, Anupam Datta, and Sharada Sundaram. Privacy and utility in business processes. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium*, pages 279–294. IEEE Computer Society, 2007. ISBN 0-7695-2819-8. doi: 10.1109/CSF.2007.26.
- [14] Kathi Fisler, Shriram Krishnamurthi, and Daniel J Dougherty. Embracing policy engineering. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, pages 109–110. ACM, 2010.

- [15] P. Swire and A. Anton. Engineers and lawyers in privacy protection: Can we all just get along? IAPP Privacy Perspectives, 2014.
- [16] Shayak Sen, Saikat Guha, Anupam Datta, Sriram K. Rajamani, Janice Tsai, and Jeannette M. Wing. Bootstrapping privacy compliance in big data systems. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, pages 327–342. IEEE Computer Society, 2014. ISBN 978-1-4799-4686-0. doi: 10.1109/SP.2014.28.
- [17] Yan Shvartzshnaider, Zvonimir Pavlinovic, Thomas Wies, Lakshminarayanan Subramanian, Prateek Mittal, and Helen Nissenbaum. The vaccine framework for building dlp systems. *arXiv preprint arXiv:1711.02742*, 2017.
- [18] Michael Carl Tschantz, Anupam Datta, and Jeannette M. Wing. Purpose restrictions on information use. In *Proceedings of the 18th European Symposium on Research in Computer Security (ESORICS)*, volume 8134 of *Lecture Notes in Computer Science*, pages 610–627. Springer Berlin Heidelberg, 2013.
- [19] Anupam Datta, Matthew Fredrikson, Gihyuk Ko, Piotr Mardziel, and Shayak Sen. Use privacy in data-driven systems: Theory and experiments with machine learnt programs. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1193–1210. ACM, 2017.
- [20] Sebastian Benthall, Anupam Datta, and Michael Carl Tschantz. Origin privacy. Report sent to DARPA as `policy-doc.pdf`, 2018. Revision of report sent to DARPA as `BenthallDattaTschantzDRAFTPLSC2017.pdf` on Aug. 22, 2017.
- [21] Judea Pearl. Probabilistic reasoning in intelligent systems: Networks of plausible inference. 1988.
- [22] Origin Privacy Team, Piotr Mardziel, Ed. `demo-scala-labels.tar.gz`. Code bundle, 2018.
- [23] Michael Veale and Reuben Binns. Fairer machine learning in the real world: Mitigating discrimination without collecting sensitive data. *Big Data & Society*, 4(2), 2017.
- [24] Daphne Koller and Brian Milch. Multi-agent influence diagrams for representing and solving games. *Games and economic behavior*, 45(1):181–221, 2003.
- [25] Origin Privacy Team, M.C. Tschantz, Ed. Gaps in brandeis presentations. Note sent Nov. 2017, 2017.
- [26] Daniel J. Solove. A taxonomy of privacy. *University of Pennsylvania Law Review*, 154(3): 477–560, 2006.
- [27] Cynthia Dwork and Moni Naor. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality*, 2(1): 93–107, 2008.

- [28] Cynthia Dwork, Frank Mcsherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.
- [29] Michael Carl Tschantz, Amit Datta, Anupam Datta, and Jeannette M. Wing. A methodology for information flow experiments. In *Computer Security Foundations Symposium*. IEEE, 2015.
- [30] Judea Pearl. *Causality*. Cambridge University Press, second edition, 2009.
- [31] Michael Carl Tschantz, Shayak Sen, and Anupam Datta. Differential privacy as a causal property. Report shared with DARPA (Oct. 13, 2017), 2017.
- [32] Michael Carl Tschantz, Shayak Sen, and Anupam Datta. Differential privacy as a causal property. *ArXiv*, 1710.05899, 2017.
- [33] John McLean. Security models and information flow. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 180–187, 1990.
- [34] James W. Gray, III. Toward a mathematical foundation for information flow security. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 21–34, 1991.
- [35] Gilles Barthe, Pedro R. D’Argenio, and Tamara Rezk. Secure information flow by self-composition. In *CSFW ’04: Proceedings of the 17th IEEE Computer Security Foundations Workshop*, page 100, 2004. ISBN 0-7695-2169-X. doi: <http://dx.doi.org/10.1109/CSFW.2004.17>.
- [36] Geoffrey Smith. Recent developments in quantitative information flow (invited tutorial). In *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 23–31. IEEE Computer Society, 2015. ISBN 978-1-4799-8875-4. doi: 10.1109/LICS.2015.13.
- [37] CE Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.
- [38] Fred I Dretske. The epistemology of belief. *Synthese*, 55(1):3–19, 1983.
- [39] Fred Dretske. *Knowledge and the Flow of Information*. MIT Press, 1981.
- [40] Christopher Meek. Strong completeness and faithfulness in bayesian networks. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 411–418. Morgan Kaufmann Publishers Inc., 1995.
- [41] Dan Geiger and Judea Pearl. Logical and algorithmic properties of independence and their application to bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 2(1-4): 165–178, 1990.
- [42] Judea Pearl. [bayesian analysis in expert systems]: Comment: Graphical models, causality and intervention. *Statistical Science*, 8(3):266–269, 1993.

- [43] James Woodward. *Making things happen: A theory of causal explanation*. Oxford university press, 2005.
- [44] Steven Sloman. *Causal models: How people think about the world and its alternatives*. Oxford University Press, 2005.
- [45] Steven A Sloman, Philip M Fernbach, and Scott Ewing. Causal models: The representational infrastructure for moral judgment. *Psychology of Learning and Motivation*, 50:1–26, 2009.
- [46] Bo Cowgill and Catherine Tucker. Algorithmic bias: A counterfactual perspective. Technical report, 2017.
- [47] Donald B Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, 66(5):688, 1974.
- [48] Geoffrey Nunberg. Farewell to the information age. *The future of the book*, pages 103–138, 1996.
- [49] Henry DeYoung, Deepak Garg, Limin Jia, Dilsun Kirli Kaynar, and Anupam Datta. Experiences in the logical specification of the HIPAA and GLBA privacy laws. In *WPES*, pages 73–82, 2010.
- [50] Office for Civil Rights (OCR), U.S. Dept. of Health and Human Services. Does the HIPAA Privacy Rule limit what a doctor can do with a family medical history? FAQ for Professions, HIPAA, 2013.
- [51] Office for Civil Rights (OCR), U.S. Dept. of Health and Human Services. Summary of the HIPAA privacy rule. Privacy for Professionals HIPAA website, 2013.
- [52] Martin Bradley and Alexander Dent. Payment card industry data security standard. 2010.
- [53] PCI Security Standards Council. Who has to comply with the pci standards? n.d. Web. 15 May 2017.
- [54] PCI Security Standards Council. Payment card industry (pci) data security standard, 2016. v3.2.
- [55] Nan Feng, Harry Jiannan Wang, and Minqiang Li. A security risk analysis model for information systems: Causal relationships of risk factors and vulnerability propagation analysis. *Information sciences*, 256:57–73, 2014.
- [56] Deepak Garg, Limin Jia, and Anupam Datta. Policy auditing over incomplete logs: Theory, implementation and applications. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pages 151–162, 2011. ISBN 978-1-4503-0948-6. doi: 10.1145/2046707.2046726.
- [57] Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.

- [58] Tore Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15:429–444, 1977.
- [59] Deirdre K. Mulligan, Colin Koopman, and Nick Doty. Privacy is an essentially contested concept: A multi-dimensional analytic for mapping privacy. *Phil. Trans. R. Soc. A*, 374 (20160118), 2016.
- [60] Amit Datta, Anupam Datta, Jael Makagon, Deirdre K Mulligan, and Michael Carl Tschantz. Discrimination in online advertising: A multidisciplinary inquiry. In *Conference on Fairness, Accountability and Transparency*, pages 20–34, 2018.
- [61] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.
- [62] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Foundations of Computer Science, 2007. FOCS’07. 48th Annual IEEE Symposium on*, pages 94–103. IEEE, 2007.
- [63] Shiva Prasad Kasiviswanathan and Adam Smith. On the ‘semantics’ of differential privacy: A bayesian formulation. *Journal of Privacy and Confidentiality*, 6(1):1–16, 2014.
- [64] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.
- [65] Paul Cuff and Lanqing Yu. Differential privacy as a mutual information constraint. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS ’16*, pages 43–54. ACM, 2016. ISBN 978-1-4503-4139-4. doi: 10.1145/2976749.2978308.
- [66] Frank McSherry. On “differential privacy as a mutual information constraint”. Blog: <https://github.com/frankmcsherry/blog/blob/master/posts/2017-01-26.md>, 2017.
- [67] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067. ACM, 2014.
- [68] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.
- [69] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 193–204. ACM, 2011.
- [70] Daniel Kifer and Ashwin Machanavajjhala. A rigorous and customizable framework for privacy. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 77–88. ACM, 2012. ISBN 978-1-4503-1248-6. doi: 10.1145/2213556.2213571.

- [71] Xi He, Ashwin Machanavajjhala, and Bolin Ding. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2014)*. ACM, 2014.
- [72] Daniel Kifer and Ashwin Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. *ACM Trans. Database Syst.*, 39(1):3:1–3:36, 2014. ISSN 0362-5915. doi: 10.1145/2514689.
- [73] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. Dependence makes you vulnerable: Differential privacy under dependent tuples. In *Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2016. doi: <http://dx.doi.org/10.14722/ndss.2016.23279>.
- [74] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, pages 75–84. ACM, 2007. ISBN 978-1-59593-631-8. doi: 10.1145/1250790.1250803.
- [75] Frank McSherry. How many secrets do you have? Blog: <https://github.com/frankmcsherry/blog/blob/master/posts/2017-02-08.md>, 2017.
- [76] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. Privacy loss in Apple’s implementation of differential privacy on MacOS 10.12. *ArXiv*, 1709.02753, 2017.

A DPI Theorems

This appendix contains proofs for several theorems extending the well-known Data Processing Inequality in information theory to configurations of random variables beyond a triplet Markov Chain. The motivation for these theorems is the desire to model information flow through a world modeled as a Bayesian network, where information flow is determined by causal flows and nomic associations. Nomic association here is measured as mutual information between two variables. The Chain rule for mutual information and the Markov properties of a Bayesian network make it possible to prove several theorems that are as far as we know new.

A.1 Triplet Structures

The Data Processing Inequality is a standard theorem in information theory. It concerns the mutual information of three variables arranged in a Markov Chain.

Definition 51. (Cover and Thomas [68]) Random variables X, Y, Z are said to *form a Markov chain in that order* (denoted $X \rightarrow Y \rightarrow Z$) if the conditional distribution of Z depends only on Y and is conditionally independent of X . Specifically, X, Y, Z form a Markov chain $X \rightarrow Y \rightarrow Z$ if the joint probability mass function can be written as

$$p(x, y, z) = p(x)p(y|x)p(z|y) \tag{11}$$

Theorem 52 (Data Processing Inequality). Given a probability model defined by the following (Markov Chain):

$$X \longrightarrow Y \longrightarrow Z$$

where $X \perp\!\!\!\perp Z|Y$, then it must be that $I(X;Y) \geq I(X;Z)$.

Proof. From [68]. By the Chain Rule, mutual information can be expanded in two different ways:

$$\begin{aligned} I(X;Y,Z) &= I(X;Z) + I(X;Y|Z) \\ &= I(X;Y) + I(X;Z|Y) \end{aligned} \quad (12)$$

Since X and Z are conditionally independent given Y , we have $I(X;Z|Y) = 0$. Since $I(X;Y|Z) \geq 0$, we have

$$I(X;Y) \geq I(X;Z) \quad (13)$$

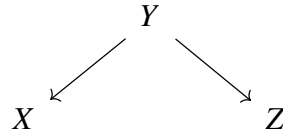
We have equality if and only if $I(X;Y|Z) = 0$ (i.e. $X \rightarrow Z \rightarrow Y$ forms a Markov Chain). Similarly, one can prove that $I(Y;Z) \geq I(X;Z)$. $\square$

Bayesian networks are a generalization of Markov chains. A Bayesian network models the probability distribution between many random variables as a directed acyclic graph. The conditional probability distribution of each variable is defined in terms of the graphical parents $pa(\cdot)$ of each variable, i.e. $P(X_i) = P(X_i|pa(X_i))$. The joint distribution is

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i|pa(X_i)) \quad (14)$$

We can now prove several theorems that are similar to the Data Processing Inequality but for other probabilistic structures besides Markov chains.

Theorem 53 (Data Sourcing Inequality). Given a probability model defined by the following (Common Cause):



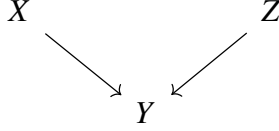
then it must be that $I(X;Y) \geq I(X;Z)$.

Proof. The implication of the common cause structure is that

$$p(x,y,z) = p(x|y)p(y)p(z|y). \quad (15)$$

It follows that $X \perp\!\!\!\perp Z|Y$. The rest of the proof is identical to the previous proof. $\square$

Theorem 54 (Unobserved common effect inequality). Given variables X, Y, Z with the common cause structure



then it must be that $I(X, Y) \geq I(X, Z) = 0$.

Proof. The implication of the structure is that

$$p(x, y, z) = p(x)p(y|x, z)p(z). \quad (16)$$

It follows that $X \perp\!\!\!\perp Z$, therefore $I(X; Z) = 0$. Because the mutual information of two variables is always nonnegative,

$$I(X, Y) \geq I(X; Z)$$

□

We note that while a similar property holds for a “collider” or common effect structure, its proof is different from the chain and common cause cases because, in general, it is not the case that $X \perp\!\!\!\perp Z|Y$ for a common effect structure. For example, when X and Z are both fair coin tosses and $Y = X \oplus Z$, X and Z are independent from each other but not when conditioned on Y .

When a common effect is in the conditioning set, the two causes depend probabilistically on each other. The extent to which these dependencies are limited can be characterized by a few equations.

Lemma 55. Given variables X_1, X_2, Y with the common effect structure $X_1 \rightarrow Y \leftarrow X_2$, then $I(X_1; X_2, Y) = I(X_1, Y|X_2)$.

Proof. By the Chain Rule for mutual information,

$$I(X_1; X_2, Y) = I(X_1; X_2) + I(X_1; Y|X_2)$$

Because of the common effect structure, $I(X_1; X_2) = 0$. Therefore, $I(X_1; X_2, Y) = I(X_1; Y|X_2)$. □

Lemma 56. Given variables X_1, X_2, Y with the common effect structure $X_1 \rightarrow Y \leftarrow X_2$, then

$$\begin{aligned} I(Y; X_1, X_2) &= I(X_1; X_2, Y) + I(X_2; Y) \\ &= I(X_2; X_1, Y) + I(X_1; Y) \end{aligned} \quad (17)$$

Proof.

$$\begin{aligned} I(X_1; X_2, Y) &= I(X_1; Y|X_2) \\ &= H(Y|X_2) - H(Y|X_1, X_2) \\ &= H(Y|X_2) - H(Y) + I(Y; X_1, X_2) \\ &= I(Y; X_1, X_2) - I(X_2; Y) \end{aligned} \quad (18)$$

which implies that

$$I(X_1; X_2, Y) + I(X_2; Y) = I(Y; X_1, X_2)$$

The proof works symmetrically for $I(X_2; X_1, Y) + I(X_1; Y) = I(Y; X_1, X_2)$ □

Lemma 57. Given variables X_1, X_2, Y with the common effect structure $X_1 \rightarrow Y \leftarrow X_2$, then $I(X_1, X_2|Y) \leq I(X_1; X_2, Y)$.

Proof.

$$\begin{aligned}
I(X_1, X_2|Y) &= H(X_1|Y) - H(X_1|X_2, Y) \\
&= H(X_1) - I(X_1; Y) - H(X_1) + I(X_1; X_2, Y) \\
&= I(X_1; X_2, Y) - I(X_1; Y) \\
&\leq I(X_1; X_2, Y)
\end{aligned} \tag{19}$$

□

Theorem 58. Given variables X_1, X_2, Y with the common effect structure $X_1 \rightarrow Y \leftarrow X_2$, then

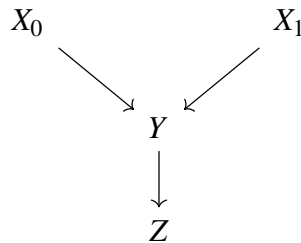
$$I(X_1, X_2; Y) \geq I(X_1; X_2, Y) = I(X_1; Y|X_2) \geq I(X_1; X_2|Y)$$

Proof. Follows from Lemmas 55, 56, and 57. □

A.2 Quartet Structures

While triplet structures (chain, common effect, and common cause) are the building blocks of larger paths in Bayesian networks, an analysis of larger, quarter structures will help us develop general theorems about the mutual information along paths.

Recall that a both with a common effect is not blocked if *either* the common effect *or* a descendant of the effect is in the conditioning set. Let's look at the following structure, which we will call a *wishbone* structure.



Here, Y is a common effect of X_0 and X_1 , and Z is a descendant of Y . How much information flows from X_0 to X_1 when Z is known?

Theorem 59. For variables X_0, X_1, Y, Z in a wishbone structure,

$$I(X_0; X_1|Z) \leq I(Y; Z)$$

Proof. Consider the quantity $I(X_0, Y; X_1, Z)$, expanded by the Chain Rule. One expansion is:

$$\begin{aligned}
I(X_0, Y; X_1, Z) &= I(Y; Z) + I(X_0; Z|Y) + I(Y; X_1|Z) + I(X_0, X_1|Y, Z) \\
&= I(Y; Z) + I(X_0, Y; X_1)
\end{aligned} \tag{20}$$

Another expansion is:

$$\begin{aligned}
I(X_0, Y; X_1, Z) &= I(X_0; Z) + I(Y; X_1 | X_0) + I(X_0; X_1 | Z) + I(Y; X_1 | X_0, Z) \\
&\geq I(Y; X_1 | X_0) + I(X_0; X_1 | Z)
\end{aligned} \tag{21}$$

By Theorem 58, we know that $I(Y; X_1 | X_0) = I(X_0, Y; X_1)$ for three variables in a common effect structure, as they are for these variables in the wishbone structure.

So we can set the two expansions equal to each other and reduce:

$$\begin{aligned}
I(Y; X_1 | X_0) + I(X_0; X_1 | Z) &\leq I(Y; Z) + I(X_0, Y; X_1) \\
I(X_0, Y; X_1) + I(X_0; X_1 | Z) &\leq I(Y; Z) + I(X_0, Y; X_1) \\
I(X_0; X_1 | Z) &\leq I(Y; Z)
\end{aligned} \tag{22}$$

□

A.3 Paths

We can now look at mutual information of nodes connected by longer paths. We start with an arbitrarily long Markov chain.

$$X_0 \longrightarrow X_1 \longrightarrow \dots \longrightarrow X_{n-1} \longrightarrow X_n$$

Theorem 60 (Chain Data Processing Inequality). Given a Markov chain of variables

$$X_1, \dots, X_n$$

such that $X_1 \rightarrow \dots \rightarrow X_n$. It must be the case that

$$I(X_1, X_n) \leq \min_i I(X_i, X_{i+1}).$$

Proof. For all i , by the Chain rule for mutual information and the independence properties of the Markov chain,

$$\begin{aligned}
I(X_0, \dots, X_i; X_{i+1}, \dots, X_n) &= \\
\sum_{j=i+1}^n I(X_0, \dots, X_i; X_j | X_{i+1}, \dots, X_j) &= \\
I(X_0, \dots, X_i; X_{i+1}) &= \\
\sum_{j=i}^1 I(X_{i+1}; X_j | X_i, \dots, X_j) &= \\
I(X_i; X_{i+1}) + \sum_{j=i-1}^1 I(X_{i+1}; X_j | X_i, \dots, X_j) &= \\
I(X_i; X_{i+1}) &
\end{aligned} \tag{23}$$

The Chain rule can expand the variables in arbitrary order. So we can also derive (using the fact that mutual information is always nonnegative):

$$\begin{aligned}
& I(X_0, \dots, X_i; X_{i+1}, \dots, X_n) \\
&= I(X_0, \dots, X_i; X_n) + \sum_{j=n-1}^{i+1} I(X_0, \dots, X_i; X_j | X_{j+1}, \dots, X_j) \\
&\geq I(X_0, \dots, X_i; X_n) \\
&= \sum_{j=0}^{n-1} I(X_n; X_j | X_{j-1}, \dots, X_0) \\
&= I(X_n; X_0) + \sum_{j=1}^{n-1} I(X_n; X_j | X_{j-1}, \dots, X_0) \\
&\geq I(X_n; X_0)
\end{aligned} \tag{24}$$

Combining these two results and generalizing across all i ,

$$\forall i, I(X_0; X_n) \leq I(X_i, X_{i+1}) \tag{25}$$

which entails that which is to be proven,

$$I(X_0; X_n) \leq \min_i I(X_i, X_{i+1}) \tag{26}$$

□

Our goal is to generalize this theorem to Bayesian paths with other structures, just found as in the previous section we found equivalents to the Data Processing Inequality in other triplet structures.

Definition 61 (Path). A *path* between two nodes X_1 and X_2 in a graph to be a sequence of nodes starting with X_1 and ending with X_2 such that successive nodes are connected by an edge (traversing in either direction).

In this section, we will only consider paths isolated from any other variables. We are interested in how to derive useful bounds on the mutual information of a path based on the mutual information of links within the path.

Definition 62 (Mutual information of a path). The *mutual information of a path* between two nodes X and Y is $I(X, Y)$.

Theorem 63 (Unobserved Path Data Processing Inequality). Given a path between X_0 and X_n of variables $X_0, \dots, X_n$, with no other connected variables. It must be the case that

$$I(X_1, X_n) \leq \min_i I(X_i, X_{i+1}).$$

Proof. This proof mirrors the proof of Theorem 60.

For any i , consider $I(X_0, \dots, X_i; X_{i+1}, \dots, X_n)$.

By the logic of Equation 23, $I(X_0, \dots, X_i; X_{i+1}, \dots, X_n) = I(X_i, X_{i+1})$.

By the logic of Equation 24, $I(X_0, \dots, X_i; X_{i+1}, \dots, X_n) \geq I(X_0, X_n)$.

Therefore, $\forall i, I(X_0; X_n) \leq I(X_i, X_{i+1})$ and $I(X_0; X_n) \leq \min_i I(X_i, X_{i+1})$. □

Theorem 63 applies to any paths on the condition that none of the variables are observed. Its proof is identical to the proof for Markov chains because isolated, unobserved paths are Markov equivalent to Markov chains.

Some proofs extending this result follow from theory of Bayesian networks. Recall that there are two conditions under which a path between two variables is blocked. First, an unobserved head-to-head connection on the path blocks the path and makes the terminal nodes conditionally independent. Second, an observation of a head-to-tail or tail-to-tail node blocks the path and makes the terminal nodes conditionally independent. If the only paths between two variables are blocked, then they are d-separated and therefore independent, with zero mutual information.

Theorem 64 (Blocked Path Mutual Information). For any blocked paths between X_0 and X_n of variables $X_0, \dots, X_n$ with no other connected variables, $I(X_0, X_n) = 0$.

Proof. If the only path between X_0 and X_n is blocked, then X_0 and X_n are d-separated and conditionally independent. If X_0 and X_n are conditionally independent, then $I(X_0, X_n) = 0$. $\square$

The difficult case for determining the mutual information of a path is the case where there are observed common effects on the paths. This breaks the conditions for the proof of Theorem 63. It is possible for $I(X_i, X_{i+1}) = 0$ but $I(X_{i-1}, X_{i+1} | X_i) > 0$. As a simple example, consider again the case where X_{i-1} and X_{i+1} are fair coin tosses and $X_i = X_{i-1} \oplus X_{i+1}$.

If there are many common effect nodes on the path and only some of them are observed, then the path is blocked and the mutual information is solved using Theorem 64; the mutual information of the path is zero. Similarly, if there are common cause or chain triplets on the path and the central node of the triplet is observed, the mutual information of the path is trivially zero. So we need consider only the case where there's a path where *all and only* the common effect nodes are observed.

Theorem 65 (Path Mutual Information Theorem (PMIT)). Given a path between X_0 and X_n of variables $\{X_0, \dots, X_n\} = \mathcal{X}$ with no other connected variables. Let $\mathcal{X}_E$ be the common effect nodes, meaning only those nodes X_i such that the edge structure of the path is $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$. The mutual information of the path when all the common effects are observed is is:

$$I(X_0; X_n | \mathcal{X}_E) \leq \min_i \begin{cases} I(X_i, X_{i+1}) & \text{if } X_i, X_{i+1} \notin \mathcal{X}_E \\ I(X_{i-1}, X_{i+1}) & \text{if } X_i \in \mathcal{X}_E \end{cases}$$

Proof. For any i , consider

$$I(X_0, \dots, X_i; X_{i+1}, \dots, X_n | \mathcal{X}_E)$$

By the Chain Rule for mutual information, this can be expanded as

$$\sum_{j=0}^i I(X_j; X_{i+1}, \dots, X_n | X_0, \dots, X_{j-1}, \mathcal{X}_E)$$

Consider two cases.

In the first case, $X_i \notin \mathcal{X}_E$ and $X_{i+1} \notin \mathcal{X}_E$.

By logic similar to Equation 23 and Equation 24, then as before $I(X_0; X_n) \leq I(X_i, X_{i+1})$.

In the second case, X_i is a common effect node, i.e $X_i \in \mathcal{X}_E$. It is not possible to have two common effect nodes adjacent on a path. So in any case where either X_{i-1} or X_{i+1} is in the

conditioning set, the path is blocked. We can therefore compute the mutual information and its Chain Rule expansion as:

$$\begin{aligned}
& I(X_0, \dots, X_{i-1}; X_{i+1}, \dots, X_n | \mathcal{X}_E) \\
&= \sum_{j=i-1}^0 I(X_j; X_{i+1}, \dots, X_n | X_{i-1}, \dots, X_{j-1}, \mathcal{X}_E) \\
&= I(X_{i-1}; X_{i+1}, \dots, X_n | \mathcal{X}_E) \\
&= I(X_{i-1}; X_{i+1} | \mathcal{X}_E) \\
&= I(X_{i-1}; X_{i+1} | X + i)
\end{aligned} \tag{27}$$

Since once again by the logic of Equation 24 this value is greater than or equal to the mutual information of the path, we have

$$I(X_0; X_n) \leq I(X_{i-1}, X_{i+1} | X_i)$$

for the cases when $X_i \in \mathcal{X}_E$.

Combining these results, we get the bound on the mutual information of the path. $\square$

Note that Theorem 63 is a special case of PMIT, or Theorem 65, where the set of common effects on the path $\mathcal{X}_E$ is empty.

B Multi-Agent Influence Diagrams and Data Games

This appendix contains formal specifics of Multi-Agent Influence Diagrams and data games.

B.1 Multi-Agent Influence Diagrams

Multi-Agent Influence Diagrams (MAIDs) are a game-theoretic extension of Bayesian networks developed by Koller and Milch [24]. A MAID is defined by:

1. A set $\mathcal{A}$ of agents
2. A set $\mathcal{X}$ of chance variables
3. A set $\mathcal{D}_a$ of decision variables for each agent $a \in \mathcal{A}$, with $\mathcal{D} = \bigcup_{a \in \mathcal{A}} \mathcal{D}_a$
4. A set $\mathcal{U}_a$ of utility variables for each agent $a \in \mathcal{A}$, with $\mathcal{U} = \bigcup_{a \in \mathcal{A}} \mathcal{U}_a$
5. A directed acyclic graph $\mathcal{G}$ that defines the parent function Pa over $\mathcal{V} = \mathcal{X} \cup \mathcal{D} \cup \mathcal{U}$
6. For each chance variable $X \in \mathcal{X}$, a CPD $Pr(X | Pa(X))$
7. For each utility variable $U \in \mathcal{U}$, a CPD $Pr(U | Pa(U))$

The decision variables represent moments where agents can make decisions about how to act given only the information provided by the variable's parents.

Definition 66 (Decision rules). A *decision rule* δ is a function that maps each instantiation $\mathbf{pa}$ of $Pa(D)$ to a probability distribution over $dom(D)$.

Definition 67 (Strategy). An assignment of decision rules to every decision $D \in \mathcal{D}_a$ for a particular agent $a \in \mathcal{A}$ is called a *strategy*.

Definition 68 (Strategy profile). An assignment σ of decision rules to every decision $D \in \mathcal{D}$ is called a *strategy profile*. A *partial strategy profile* $\sigma_{\mathcal{E}}$ is an assignment of decision rules to a subset $\mathcal{E} \subset \mathcal{D}$. $\sigma_{-\mathcal{E}}$ refers to a restriction of σ to variables not in $\mathcal{E}$.

Decision rules are of the same form as CPDs, and so a MAID can be transformed into a Bayes network by replacing every decision variable with a random variable with the CPD of the decision rule of a strategy profile.

Definition 69. If $\mathcal{M}$ is a MAID and σ is a strategy profile for $\mathcal{M}$, then the *joint distribution* for $\mathcal{M}$ induced by σ , denoted $P_{\mathcal{M}[\sigma]}$, is the joint distribution over $\mathcal{V}$ defined by the Bayes net where:

- the set of variables is $\mathcal{V}$;
- for $X, Y \in \mathcal{V}$, there is an edge $X \rightarrow Y$ if and only if $X \in Pa(Y)$;
- for all $X \in \mathcal{X} \cup \mathcal{U}$, the CPD for X is $Pr(X)$;
- for all $D \in \mathcal{D}$, the CPD for D is $\sigma(D)$.

Definition 70. Let $\mathcal{E}$ be a subset of $\mathcal{D}_a$ and let σ be a strategy profile. We say that $\sigma_{*\mathcal{E}}$ is *optimal for the strategy profile* σ if, in the induced MAID $\mathcal{M}[\sigma_{-\mathcal{E}}]$, where the only remaining decisions are those in $\mathcal{E}$, the strategy $\sigma_{*\mathcal{E}}$ is optimal, i.e., for all strategies $\sigma'_{\mathcal{E}}$:

$$EU_a((\sigma_{-\mathcal{E}}, \sigma_{*\mathcal{E}})) \geq EU_a((\sigma_{-\mathcal{E}}, \sigma'_{\mathcal{E}}))$$

A major contribution of Koller and Milch [24] is their analysis of how to efficiently discover Nash Equilibrium strategy profiles for MAIDs. Their method involves analyzing the qualitative graphical structure of the MAID to discover the *strategic reliance* of decision variables. When a decision variable D strategically relies on D' , then in principle the choice of the optimal decision rule for D depends on the choice of the decision rule for D' .

Definition 71 (Strategic reliance). Let D and D' be decision nodes in a MAID $\mathcal{M}$. D *strategically relies on* D' if there exist two strategy profiles σ and σ' and a decision rule δ for D such that:

- δ is optimal for σ ;
- σ' differs from σ only at D' ;

but no decision rule δ^* that agrees with δ on all parent instantiations $\mathbf{pa} \in dom(Pa(D))$ where $P_{\mathcal{M}[\sigma]}(\mathbf{pa}) > 0$ is optimal for σ' .

Definition 72 (s-reachable). A node D' is *s-reachable* from a node D in a MAID $\mathcal{M}$ if there is some utility node $U \in \mathcal{U}_D$ such that if a new parent $\hat{D}'$ were added to D' , there would be an active path in $\mathcal{M}$ from $\hat{D}'$ to U given $Pa(D) \cup \{D\}$, where a path is active in a MAID if it is active in the same graph, viewed as a BN.

Theorem 73. If D and D' are two decision nodes in a MAID $\mathcal{M}$ and D' is not s-reachable from D in $\mathcal{M}$, then D does not strategically rely on D' .

B.1.1 Tactical independence

This dissertation introduces a new concept related to Multi-Agent Influence Diagrams: tactical independence.

Definition 74 (Tactical independence). For decision variables D and D' in MAID $\mathcal{M}$, D and D' are *tactically independent* for conditioning set $\mathcal{C}$ iff for all strategy profiles σ on $\mathcal{M}$, in $P_{\mathcal{M}[\sigma]}$, the joint distribution for $\mathcal{M}$ induced by σ ,

$$D \perp\!\!\!\perp D' | \mathcal{C}$$

Because tactical independence depends on the independence of variables on an induced probability distribution that is representable by a Bayesian network, the d-separation tests for independence apply readily.

Theorem 75. For decision variables D and D' in MAID $\mathcal{M}$, and for conditioning set $\mathcal{C}$, if D and D' are d-separated given $\mathcal{C}$ on $\mathcal{M}$ considered as a Bayesian network, then D and D' are tactically independent given $\mathcal{C}$.

Proof. Suppose D and D' are d-separated given $\mathcal{C}$ on $\mathcal{M}$ considered as a Bayesian network.

For any strategy profile σ , the joint distribution for $\mathcal{M}$ induced by σ , $P_{\mathcal{M}[\sigma]}$ has the same graphical structure as $\mathcal{M}$ considered as a Bayesian network.

Therefore, D and D' are d-separated given $\mathcal{C}$ in the graph corresponding to $P_{\mathcal{M}[\sigma]}$ for all σ .

Because D and D' are d-separated given $\mathcal{C}$ in the Bayesian network, $D \perp\!\!\!\perp D' | \mathcal{C}$. $\square$

B.1.2 Notation

We will use a slightly different graphical notation than that used by Koller and Milch [24].

In the models in this paper, we will denote random variables with undecorated capital letters, e.g. A, B, C . I will denote strategic nodes with a tilde over a capital letter, e.g. $\tilde{A}, \tilde{B}, \tilde{C}$. The random variable defined by the optimal strategy at a decision node, when such a variable is well-defined, will be denoted with a hat, e.g. $\hat{A}, \hat{B}, \hat{C}$. Nodes that represent the payoff or utility to an agent will be denoted with a breve, e.g. $\breve{A}, \breve{B}, \breve{C}$. Particular agents will be identified by a lower case letter and the assignment of strategic and utility nodes to them will be denoted by subscript. E.g., $\tilde{A}_q$ and $\breve{U}_q$ denote an action taken by agent q and a payoff awarded to q , respectively.

B.2 Data Games

What distinguishes a data game from a MAID is the use of optional arrows to support mechanism design. A dotted arrow in a data game an optional arrow. The diagram defines two separate models, one including the arrow and one without. When considering an instantiation of the model with the dotted edge present, we will say the model or edge is *open*. When the edge is absent, we'll say it's *closed*.

As we have distinguished between strategic reliance and tactical independence, we can distinguish between the strategic and tactical value of information.

The strategic value of an information flow to an agent is the difference in utility to that agent in the open and closed conditions of the game, given each game is at strategic equilibrium for all players.

Definition 76 (Strategic value of information). Given two MAID diagrams $\mathcal{M}_o$ and $\mathcal{M}_c$ that differ only by a single edge, e , and a strategic profile solution for each diagram, $\hat{\sigma}_o$ and $\hat{\sigma}_c$, the *strategic value of e to a* is the difference in expected utility to a under the two respective induced joint distributions:

$$E(P_{\mathcal{M}_o[\hat{\sigma}_o]}(U_a)) - E(P_{\mathcal{M}_c[\hat{\sigma}_c]}(U_a))$$

Definition 76 is an incomplete definition because it leaves open what *solution concept* is used to determine the strategic profile solutions. For the purpose of the results in this paper, we use Nash Equilibrium as the solution concept for determining strategic value of information.

In contrast with the strategic value of information, the tactical value of information is the value of the information to an agent given an otherwise fixed strategy profile. We allow the agent receiving the data to make a tactical adjustment to their strategy at the decision variable at the head of the new information flow.

Definition 77 (Best tactical response to information). Given two MAID diagrams $\mathcal{M}_o$ and $\mathcal{M}_c$ differing only in optional edge e with head in decision variable D_a , the *best tactical response to e* given strategy profile solution $\hat{\sigma}$, $\hat{\delta}_{\sigma,e}$ is the decision rule δ for D such that δ is optimal for $\hat{\sigma}$ for player a .

Definition 78 (Tactical value of information). Given two MAID diagrams $\mathcal{M}_o$ and $\mathcal{M}_c$ differing only in optional edge e with head in decision variable D , the *tactical value of e to agent a* given strategy profile solution $\hat{\sigma}$ is the difference in expected utility of the open condition with the best tactical response to e and the closed condition using the original strategy:

$$EU_a((\hat{\sigma}_{-D}, \hat{\delta}_{\hat{\sigma},e}) - EU_a(\hat{\sigma}))$$

Note that the uniqueness of a best tactical response has not yet been proven. However, if the best tactical response is not unique, then the tactical value of the information will be the same for any best tactical response. This definition, like Definition 76, depends on an implicit solution concept.

C Gaps in Brandeis Presentations

The goal of the Brandeis project is “enabling safe and predictable sharing of data in which privacy is preserved”. This goal requires an understanding of what is safe and predictable with respect to privacy, which in turn requires attention to the concept of privacy. As far as we can tell from the presentations made by the Brandeis teams, many are largely attempting to preserve privacy through two technical approaches: (1) using variations of differential privacy and (2) identifying sensitive information for special treatment (e.g., processing by differential privacy), or, more generally, relying upon policies to identify what to protect. (A noteworthy exception is CMU’s efforts to infer user privacy preferences.) We have not seen much attention paid to whether these techniques actually preserve privacy, leaving key questions unanswered.

Do these variations of differential privacy actually preserve privacy? A starting point for answering this question is determining whether differential privacy preserves privacy. While differential privacy is sometimes taken to be a guarantee of privacy, it is just a mathematical property that an algorithm may or may not have. Privacy is not merely a technical requirement. Such requirements must be justified by socially meaningful analyses of privacy. Such an analysis is subtle for differential privacy. Despite popular impressions, it does not guarantee that nothing is learned about the data subjects, but rather only that what will be learned would be learned regardless of any one data subject's participation. This has led some to question whether differential privacy provides privacy at all (e.g., [69, 70, 71, 72, 73]). A privacy analysis could examine whether these concerns, or others, apply to the Brandeis use cases.

Additional concerns come from the variations of differential privacy used by some Brandeis projects. For example, one project uses local, instead of global, sensitivity [74]. The privacy provided by this definition is far more subtle than differential privacy [63]. Even the commonly accepted variation called (ϵ, δ) -differential privacy gives pause to one of the creators of the original, ϵ -differential privacy [75].

Further issues can arise in the details of how differential privacy is deployed. For example, recent work has found that Apple's deployment of differential privacy appears to set, and reset, the privacy budget ϵ in a way that fails to actually protect privacy [76].

How can one determine what information is sensitive? As far as we can tell from the presentations of some Brandeis teams, the classification of information as either sensitive or insensitive is done outside their systems and accepted as given and infallible. Since the technical methods being developed only preserve privacy to the extent to which such classifications are in fact correct, pushing this determination outside of the technical systems makes them subject to abusive use and raises questions about how they can or should be used in actual deployments. Indeed, in a sense, punting on such a determination is begging the question: these systems claim to preserve privacy by protecting sensitive information, but "sensitive information" is just a rewording of "private information" with no instructions on how to identify it. This is no more helpful than being told to protect what must be protected. Furthermore, given the rejection of the sensitive/insensitive dichotomy by Nissenbaum's Contextual Integrity [4], we worry that making this determination is meaningless and that protecting privacy requires making more nuanced determinations.

A more complex version of this issue appears when projects plan to enforce policies without explaining how the policies will be crafted. Given that crafting policies requires making judgement calls, it is tempting to simply appeal to the authority of some policymaker as providing the policy as a given. However, such appeals to authority are just that. To actually show that a policy preserves privacy requires a well reasoned argument from social factors. Without such grounding, policies can slip toward the current status quo for online services, "privacy policies" in name only that do nothing to preserve privacy and instead serve as waivers of one's rights.

Making such determinations and judgement calls might appear outside the scope of technical work. Such a view kicks the can down the road to policymakers, lawyers, or practitioners attempting to deploy technical systems. These workers often have a weaker understanding of the consequences of policy decisions than the researchers developing the systems. Not only can this differential appear due to the researchers' deeper understanding of their technology and its configuration, but also because researchers typically have a better understanding of the design consider-

ations that lead to their approach in the first place and its intended use. While researchers cannot presume to have all the answers, they should at least provide guidance on the appropriate selection of policies in the form of explaining their consequences and tradeoffs with respect to normative views of privacy.

D Publications

Michael Carl Tschantz, Shayak Sen, and Anupam Datta (2017). “Differential Privacy as a Causal Property”, ArXiv 1710.05899 (2017). Shared with DARPA on Oct. 13, 2017.

Sebastian Benthall, Seda Gürses and Helen Nissenbaum (2017), ”Contextual Integrity through the Lens of Computer Science”, Foundations and Trends® in Privacy and Security: Vol. 2: No. 1, pp 1-69. <http://dx.doi.org/10.1561/33000000016>

E Unpublished Reports

Sebastian Benthall, Anupam Datta, and Michael Carl Tschantz (2017). “Toward Origin Privacy”. Discussed at the 10th Annual Privacy Law Scholars Conference. Sent to DARPA as Benthall-DattaTschantzDRAFTPLSC2017.pdf on Aug. 22, 2017. Revision sent with this final report as policy-doc.pdf.

Michael Carl Tschantz, ed. (2017). “A Policy Language for Origin Privacy”. Sent to DARPA as spec-lang.pdf on Oct. 13, 2017. Revision sent with this final report as spec-lang-enforce.pdf.

Michael Carl Tschantz, ed. (2017), “Gaps in Brandeis Presentations”. Note sent to Joshua Baron as gaps.pdf on Nov. 12, 2017.

Michael Carl Tschantz, ed. (2017), “An Example of Origin Privacy”. Note sent to Joshua Baron as example.pdf on Nov. 12, 2017.

Michael Carl Tschantz, ed. (2018). “A Policy Language for Origin Privacy with Enforcement”. Sent to DARPA as spec-lang-enforce.pdf with this final report. A revised version of spec-lang.pdf.

Sebastian Benthall, Anupam Datta, and Michael Carl Tschantz (2018). “Origin Privacy: Causation and Data Protection.” Sent to DARPA as tech-doc.pdf with this final report.

Sebastian Benthall, Anupam Datta, and Michael Carl Tschantz (2018). “Origin Privacy”. Sent to DARPA as policy-doc.pdf with this final report. A revised version of BenthallDattaTschantz-DRAFTPLSC2017.pdf.