ACCOUNTABLE ALGORITHMS

Joshua A. Kroll, Joanna Huey, Solon Barocas, Edward W. Felten, Joel R. Reidenberg, David G. Robinson, and Harlan Yu

165 UNIV. OF PENN. L. REV. --- (forthcoming 2017)

Abstract

Many important decisions historically made by people are now made by computers. Algorithms count votes, approve loan and credit card applications, target citizens or neighborhoods for police scrutiny, select taxpayers for an IRS audit, and grant or deny immigration visas.

The accountability mechanisms and legal standards that govern such decision processes have not kept pace with technology. The tools currently available to policymakers, legislators, and courts were developed to oversee human decision-makers and often fail when applied to computers instead: for example, how do you judge the intent of a piece of software? Additional approaches are needed to make automated decision systems—with their potentially incorrect, unjustified or unfair results—accountable and governable. This Article reveals a new technological toolkit to verify that automated decisions comply with key standards of legal fairness.

We challenge the dominant position in the legal literature that transparency will solve these problems. Disclosure of source code is often neither necessary (because of alternative techniques from computer science) nor sufficient (because of the complexity of code) to demonstrate the fairness of a process. Furthermore, transparency may be undesirable, such as when it permits tax cheats or terrorists to game the systems determining audits or security screening.

The central issue is how to assure the interests of citizens, and society as a whole, in making these processes more accountable. This Article argues that technology is creating new opportunities—more subtle and flexible than total transparency—to design decision-making algorithms so that they better align with legal and policy objectives. Doing so will improve not only the current governance of algorithms, but also—in certain cases—the governance of decision-making in general. The implicit (or explicit) biases of human decision-
makers can be difficult to find and root out, but we can peer into the “brain” of an algorithm: computational processes and purpose specifications can be declared prior to use and verified afterwards.

The technological tools introduced in this Article apply widely. They can be used in designing decision-making processes from both the private and public sectors, and they can be tailored to verify different characteristics as desired by decision-makers, regulators, or the public. By forcing a more careful consideration of the effects of decision rules, they also engender policy discussions and closer looks at legal standards. As such, these tools have far-reaching implications throughout law and society.

Part I of this Article provides an accessible and concise introduction to foundational computer science concepts that can be used to verify and demonstrate compliance with key standards of legal fairness for automated decisions without revealing key attributes of the decision or the process by which the decision was reached. Part II then describes how these techniques can assure that decisions are made with the key governance attribute of procedural regularity, meaning that decisions are made under an announced set of rules consistently applied in each case. We demonstrate how this approach could be used to redesign and resolve issues with the State Department’s diversity visa lottery. In Part III, we go further and explore how other computational techniques can assure that automated decisions preserve fidelity to substantive legal and policy choices. We show how these tools may be used to assure that certain kinds of unjust discrimination are avoided and that automated decision processes behave in ways that comport with the social or legal standards that govern the decision. We also show how algorithmic decision-making may even complicate existing doctrines of disparate treatment and disparate impact, and we discuss some recent computer science work on detecting and removing discrimination in algorithms, especially in the context of big data and machine learning. And lastly in Part IV, we propose an agenda to further synergistic collaboration between computer science, law and policy to advance the design of automated decision processes for accountability.
Table of Contents

I. How Computer Scientists Build and Evaluate Software.. 9
   A. Assessing Computer Systems ........................................... 9
   B. The Importance of Randomness ..................................... 12

II. Designing Computer Systems for Procedural Regularity ........................... 14
   A. Transparency and its Limits ....................................... 15
   B. Auditing and its Limits ............................................... 17
   C. Technical Tools for Procedural Regularity ................ 18
      1. Cryptographic Commitments .................................. 18
      2. Zero-Knowledge Proofs .......................................... 21
      3. Fair Random Choices .............................................. 22
   D. Applying Technical Tools to Reform the Diversity Visa Lottery ........................................ 25
      1. Current DVL Procedure .......................................... 25
      2. Transparency Is Not Enough .................................. 26
      3. Designing the DVL for Accountability ................... 26

III. Designing Algorithms to Assure Fidelity to Substantive Policy Choices ........................................... 28
   A. Possible Discriminatory Effects of Algorithms .......... 30
   B. Technical Tools for Non-Discrimination ................... 35
      1. Learning from Experience ...................................... 35
      2. Fair Machine Learning ............................................ 36
      3. Discrimination, Data Use, and Privacy ................. 42
      4. Zero-knowledge Proofs .......................................... 44
   C. Anti-Discrimination Law and Algorithmic Decision-Making ........................................... 45
      2. Ricci Impels Designing for Non-Discrimination .... 48

IV. Fostering Collaboration Across Computer Science, Law, and Policy.......................................................... 49
   A. Recommendations for Computer Scientists: Design for After-the-Fact Oversight .......................................................... 50
   B. Recommendations for Law- and Policymakers ........... 53
      1. Reduced Benefits of Ambiguity ................................. 54
Many important decisions historically made by people are now made by computer systems: votes are counted; voter rolls are purged; loan and credit card applications are approved\(^1\) welfare and financial aid decisions are made;\(^2\) taxpayers are chosen for audits; citizens or neighborhoods are targeted for police scrutiny;\(^3\) air travelers are selected for search;\(^4\) visas are granted or denied. The efficiency and accuracy of automated decision-making ensures that its domain will continue to expand. Even mundane activities now involve complex computerized decisions: everything from cars to home appliances now regularly execute computer code as part of their normal operations.

However, the accountability mechanisms and legal standards that govern decision processes have not kept pace with technology. The tools currently available to policymakers, legislators, and courts were developed primarily to oversee human decision makers. Many observers have argued that our current frameworks are not well adapted for situations in which a potentially incorrect,\(^5\)

---

\(^1\) See, e.g. Calyx Software, Automated Underwriting System, http://www.calyxsoftware.com/products/CalyxDecisioningSystems/calyxAUS.asp (Calyx offers clients an automated underwriting system to vet loan applications for approval against pre-determined guidelines.)


\(^3\) See David Robinson, Harlan Yu & Aaron Rieke, Civil Rights, Big Data and Our Algorithmic Future 18-19 (2014), http://bigdata.fairness.io/wp-content/uploads/2014/11/Civil_Rights_Big_Data_and_Our_Algorithmic-Future_v1.1.pdf (Describing the Chicago Police Department’s ‘‘Custom Notification Program,’ which sends police (or sometimes mails letters) to peoples’ homes to offer social services and a tailored warning.”)

\(^4\) See Secure Flight Records — Notice of modified Privacy Act System of Records, 78 Fed. Reg. 175, 55,270, 55,271 (Sept. 10, 2013) (“the passenger prescreening computer system will conduct risk-based analysis of passenger data . . . TSA will then review this information using intelligence-driven, risk-based analysis to determine whether individual passengers will receive expedited, standard, or enhanced screening[].”)

\(^5\) Danielle Citron, Technological Due Process, 85 Wash. Univ. L. Rev. 1249-1313 (2007) [hereinafter “Technological Due Process”].
unjustified, or unfair outcome emerges from a computer. Citizens, and society as a whole, have an interest in making these processes more accountable. If these new inventions are to be made governable, this gap must be bridged.

In this paper, we describe how compliance with key standards of legal fairness for automated decisions can be verified and demonstrated, both to the public at large and to competent oversight bodies. Such compliance can be established without revealing private attributes either of decisions themselves or the process by which decisions are reached. We acknowledge two approaches to verifying compliance: ex ante approaches to determining whether a decision process is fair, adequate, or correct, as are commonly studied by computer scientists; and ex post approaches such as review and oversight which are common in existing governance structures. Our proposals aim to use the tools of the first approach to guarantee that the second approach can function effectively.

We begin with an accessible and concise introduction to computer science concepts on which our argument relies. We then describe how these techniques can assure that decisions are made under an announced set of rules consistently applied in each case, a condition we call procedural regularity. The techniques we describe can be applied further to demonstrate compliance with substantive rudimentary policy choices, such as blindness to a particular attribute like the use of race in credit decisions or the requirement that a certain class of analysis be applied for certain decisions. We expand our discussion to explore how other computational techniques can assure that automated decisions avoid certain kinds of unjust discrimination. And finally, we propose next steps to further critically important collaboration between computer scientists and policymakers.

Legal scholars have argued for twenty years that automated processing requires more transparency, but it is far from obvious

---

6 Id.


8 See, e.g., Citron, supra note 5; Paul M. Schwartz, Data Processing and Government Administration: The Failure of the American Legal Response to the Computer, 43 Hastings L. J. 1321, 1323 & 1325 (1992)(“So long as government bureaucracy relies on the technical treatment of personal information, the law must pay attention to the structure of data processing... There are three essential elements to this response: structuring transparency of data processing systems; granting limited procedural and substantive rights ...; creating independent government monitoring of data processing systems.”)
what form such transparency should take. Perhaps the most obvious approach is to disclose a system’s source code, but this is at best a partial solution to the problem of accountability for automated decisions. The source code of computer systems is legible only to experts, and even they often struggle to understand what it will do: inspecting source code is a very limited way of predicting how a computer program will behave. In systems that use a popular approach to automated decision-making called machine learning, the decisional rule itself may emerge from the specific data under analysis, in ways that no human can explain. In this case, source code alone teaches a reviewer very little.

Moreover, in many of the instances people care about, full transparency will not be possible. The process for deciding which tax returns to audit, or whom to pull aside for secondary security screening at the airport, may need to be partly opaque to prevent tax cheats or terrorists from gaming the system. When the decision being regulated is a commercial one, such as an offer of credit, transparency may be undesirable because it defeats the legitimate protection of commercially proprietary information or trade secrets. Finally, when an explanation of how a rule operates requires disclosing the data under analysis and those data are private or sensitive (e.g., in adjudicating a commercial offer of credit, a lender reviews detailed financial information about the applicant), disclosure of the data may be undesirable or even legally barred.

And making the rule transparent—whether through source code disclosure or otherwise—may still fail to resolve the concerns of many participants. No matter how much transparency surrounds a rule, people can still wonder whether that rule was actually used to reach decisions in their own case: Particularly where an element of randomness is involved in the process, a person audited or patted down may wonder: Was I really chosen by the decision policy or has some bureaucrat singled me out on a whim? But full disclosure of how particular decisions were reached is often unattractive because the decisions incorporate sensitive health, financial or other private information either as input, as output, or both (for example, an individual’s tax audit status may be sensitive

---

9 See discussion of static analysis infra.
10 See Stanford University, Coursera: Machine Learning Course, https://www.coursera.org/learn/machine-learning/home/info (“Machine learning is the science of getting computers to act without being explicitly programmed.”)
or protected on its own, but it may also imply details about that individual’s financial data).

Fortunately, technology is creating new opportunities—more subtle and flexible than total transparency—to make automated decision making more accountable to legal and policy objectives. Although the current governance of automated decision making is underdeveloped, computerized processes can be designed for governance and accountability. Doing so will improve not only the current governance of computer systems, but also—in certain cases—the governance of decision-making in general. The implicit (or explicit) biases of human decision-makers can be difficult to find and root out, but we can peer into the “brain” of a computerized decision process: computational processes and purpose specifications can be declared prior to use and verified afterwards. As such computer science methods used in the design of computer systems can play an important role in enhancing governability.

In order for a human process or a traditional bureaucracy to function in an accountable way, accountability must be part of the system’s design from the start. This Article argues that the same is true for computer systems that fulfill important functions. Designers of such systems—and the non-technical stakeholders who often oversee or control system design—must begin with oversight and accountability in mind. We offer examples of current technological tools that could aid in that design, as well as suggestions for dealing with the apparent mismatch between policy ambiguity and technical precision.

In Part I of this Article, we provide an accessible introduction to how computer scientists build and evaluate computer systems and the software and algorithms \(^{11}\) that comprise them. In particular, we describe how computer scientists evaluate a program to verify that it has desired properties, and discuss the value of randomness in the construction of many computer systems.

Part II examines how to design computer systems for procedural regularity which is a key governance attribute enshrined in law and policy. We consider how participants and observers can

---

\(^{11}\) In this article, we limit our use of the word “algorithm” to its usage in computer science, where it refers to a well-defined set of steps for accomplishing a certain goal. In other contexts, where other authors use the term algorithm, we describe automated decision processes reflecting decision policies implemented by pieces of software all comprising computer systems. Our adoption of the phrase “computer systems” was suggested by (and originally due to) Helen Nissenbaum and we are grateful for the precision it provides.
be assured that each individual decision was made according to the same procedure (for example, how observers can be assured that the decision-maker is not choosing outcomes on a whim while merely claiming to follow an announced rule). We describe why disclosure of a piece of source code can be impractical or insufficient for these ends. We then focus on tools that can communicate partial information about secret processes, so that accountability and oversight continue to function even when policy interests, personal privacy, trade secrets or other concerns protect a computer system, a piece of software, its inputs, or its outputs from disclosure. Part II also describes how these same techniques can be used to demonstrate basic properties of a secret rule above and beyond procedural regularity, namely substantive choices (for example, the property that a secret decision rule is blind to a particular protected attribute or that the rule fits a particular type or pattern). Putting it all together, we provide an illustrative example of how to redesign an existing, legally-mandated automated decision making system—the State Department’s Diversity Visa Lottery—so that it is provably accountable.

Part III considers the broader question of how to assess whether automated decision systems treat people (including protected groups) in ways that comport with the social or legal standards that govern the decision being made. We explore the discriminatory effect that automated decision making can have, noting real-world examples of key hazards and the newfound risks posed by the transition from human-mediated decision making to automated decision making. We further describe how automated decision-making may complicate existing doctrines of disparate treatment and disparate impact. We discuss some recent computer science work on detecting and removing discrimination in computer systems, especially in the context of big data and machine learning.

Part IV concludes by calling for increased collaboration between computer scientists and legal policymakers to develop and apply technical tools for the governance of computer systems.

12 A concrete example would be the requirement that some decision only accounts for certain information for certain purposes, as in a system for screening job applicants which is allowed to take as input the gender of applicants, but only for the purpose of keeping informational statistics and not for making screening decisions.

13 Note that this type of evaluation depends upon having already verified procedural regularity. If it cannot be determined that a particular algorithm was used to make a decision, it is fruitless to try to verify properties of that algorithm.
Given the ever-widening reach of fully automated decisions, it is essential for computer scientists to know which tools need to be developed and for policymakers to know what technologies are already available. We offer recommendations for bridging the gap between technologists’ desire for specification and the policy process’s need for ambiguity. As a first step, we urge policymakers to recognize that accountability is feasible even when the details of a computer system must be kept secret. We also argue that the ambiguities, contradictions, and uncertainties of the policy process need not discourage computer scientists from engaging constructively in it.

I. HOW COMPUTER SCIENTISTS BUILD AND EVALUATE SOFTWARE

This section provides a brief and accessible map of key concepts, and also offers some insight into how computer scientists think about and approach these challenges.

A. Assessing Computer Systems

In general, a computer program is something that takes a set of inputs and produces a set of outputs. Programmers often structure or design programs with an eye towards evaluation and testing. Many respected and popular approaches to software engineering are based on this principle. The programmer can write the program in ways that lend themselves to analysis. For example, the programmer can:

- Organize the code into modules that can be evaluated separately and then combined
- Annotate the code with “assertions,” simple statements about the code that describe error conditions under which the program should crash immediately. Because they cause the program to crash, assertions are intended to be true if the program is running as expected. Assertion failures when a program is run can show a programmer that their assumptions about the internal state or the environment of their program were incorrect.
- Provide a detailed description specifying the program’s behavior along with a machine-checkable proof that the code satisfies this specification. The third design choice is

---

14 In particular, Test Driven Development (TDD) is a software engineering methodology practiced by many major software companies. See e.g. Beck, K. Test-Driven Development by Example, Addison Wesley - Vaseem, 2003
the most helpful thing a programmer can do to facilitate testing because verification of such proofs is easier than might be expected.\textsuperscript{15}

This list of techniques is deliberately partial and illustrative.

Computer scientists evaluate programs using static methods, which look at the code without running the program, and dynamic methods, which run the program and assess the outputs for particular inputs. Dynamic methods can be divided into 1) observational methods in which an analyst can see how the program runs in the field with its natural inputs and 2) testing methods, which are more powerful, where an analyst chooses inputs and submits them to the program.

Each approach has its weaknesses. Static methods cannot always predict what a program will do. Code can be complicated, and even expert analysis often misses eventual problems with the behavior of the program. For example, the Heartbleed security flaw was a potentially catastrophic vulnerability for most internet users that was caused by a common programming error—but that error made it through an open source vetting process and then sat unnoticed for two years, even though anyone was free to read and analyze the code during that time. \textsuperscript{16} Even best-of-breed commercial solutions for discovering problems in software code did not find the Heartbleed bug. This experience underscores how

\textsuperscript{15} A simple example is a technique called model checking, usually applied to computer hardware designs, in which the property desired and the hardware or program are represented as logical formulae and an automated tool performs exhaustive search to check whether those formulae are not consistent. See e.g. Model Checking, Edmund M. Clarke, Jr., Orna Grumberg and Doron A. Peled, MIT Press, 1999. An even simpler example comes from the concept of types in programming languages, which associate the data values the program operates on into descriptive classes and provide rules for how those classes should interact. For example, it should not be possible to add mathematically a number like “42” to a string of text like “Hello, World!”. Because both kinds of data are represented inside the computer as bits and bytes, without a type system, the computer would be free to try this nonsensical behavior, which might lead to bugs. Type systems help programmers avoid mistakes and can express extremely complex relationships among the data processed by the program. For more information, see e.g., Benjamin Pierce, Types and Programming Languages, MIT Press: 2002

difficult it can be to find even small and simple mistakes. More complex errors can evade scrutiny even more easily.\textsuperscript{17}

However, static methods can be very useful in establishing facts about a program such as the nature of the data it takes in, class of output it can produce, the general shape of the program, and the technologies involved in the program’s implementation. Advanced analysis can, in some cases, determine some aspects of programs’ behaviors and establish program \textit{invariants}, or facts about the program’s behavior which are true regardless of what input data the program receives. Programs which are specially designed to take advantage of more advanced analysis techniques can use static methods to prove formally complex invariants about their behavior. These techniques have been deployed in the aviation industry, for example, to ensure that the software which provides guidance functionality on rockets, airplanes, satellites, and scientific probes does not ever crash, as software failures have caused the losses of several vehicles in the past.\textsuperscript{18}

While static methods may miss revealing what a program will do, dynamic methods are limited by the finite number of inputs that can be tested or outputs that can be observed. This is important because decision policies tend to have many more possible inputs than a dynamic analysis can observe or test. Dynamic methods can explore only a small subset of those potential inputs.\textsuperscript{19} For most

\textsuperscript{17} There are some surprising limitations to the ability to evaluate code statically. The halting problem is one example: Alan Turing proved that there is no single algorithm that can predict whether, for any given program and input, the program will finish running at some point (halt) or will run forever. See Turing, A. “On computable numbers, with an application to the Entscheidungsproblem,” Proceedings of the London Mathematical Society, Series 2, Volume 42 (1937), pp 230–265 and Turing, A. “On Computable Numbers, with an Application to the Entscheidungsproblem. A Correction,” Proceedings of the London Mathematical Society, Series 2, Volume 43 (1938), pp 544–546.

\textsuperscript{18} Both the Ariane V and Mars Polar Lander crashed due to software failures. Similarly a software configuration error caused the crash of an Airbus A400M military transport. See Sean Gallagher, Airbus confirms software configuration issue caused plane crash, Ars Technica, June 1, 2015, http://arstechnica.com/information-technology/2015/06/airbus-confirms-software-configuration-error-caused-plane-crash/

\textsuperscript{19} Even auditing techniques that involve significant automation may not be able to cover the full range of possible input data if that range cannot be limited in advance to a small enough size to be searched effectively. For programmers testing their own software, achieving complete coverage of a program’s behavior by testing alone is considered impossible. Indeed, if testing for the correct behavior of a program were possible at a modest cost, then there would be no bugs in modern software. For a formal version of this argument, see Rice,
policies used in decision making, the observed or tested input-output pairs may be explained by many different hypotheses about the behavior of the program. In other words, the actual policy choice cannot be inferred from the inputs and outputs, even after a great deal of testing. No amount of dynamic testing can make an observer certain that he or she knows what the computer would do in some other situation that has yet to be tested.  

While both static and dynamic methods are after-the-fact assessments—they take the computer system and its design as a given—using both approaches together is often helpful. If an analyst can establish through static methods that a program behaves identically over some class of inputs, the analyst can test a single input from that class and infer the program behavior for the rest of the class. However, not every computer program will be able to be fully analyzed, even with such a combination of methods.  

B. The Importance of Randomness

Randomness is essential to the design of many computer systems. However, when it is used, accountability can easily be


20 Computer security experts often worry about so-called “back doors”, which are unnoticed modifications to software that cause it to behave in unexpected, malicious ways when presented with certain special inputs known only to an attacker. There are even annual contests in which the organizers “propose a challenge to coders to solve a simple data processing problem, but with covert malicious behavior. Examples include miscounting votes, shaving money from financial transactions, or leaking information to an eavesdropper. The main goal, however, is to write source code that easily passes visual inspection by other programmers.” The Underhanded C Contest. http://www.underhanded-c.org/.

Back doors have been discovered sitting undetected for many years in commercial security-focused infrastructure products subject to significant expert review, including the Juniper NetScreen line of devices. See Green, M.D. “On the Juniper Backdoor” 22 Dec. 2015. http://blog.cryptographyengineering.com/2015/12/on-juniper-backdoor.html

21 The effectiveness of program analysis is strongly bounded by Rice’s Theorem, which says that for any non-trivial property of program behavior, there cannot exist an algorithm that takes a program and determines whether its behavior has that property; any such algorithm must get some cases wrong even if the algorithm can do both static and dynamic analyses of the program. See supra, note 12.

22 In fact, there is suggestive theoretical evidence that the power of randomness may be fundamental: there are problems for which the best known randomized algorithm performs much better than the best known deterministic algorithm. For example, the well-studied “multi-armed bandit” problem in statistics has seen wide application in the field of machine learning, where randomized
lost, since by definition any outcome which a randomized process could have produced is at least facially consistent with the design of that process.\textsuperscript{23} Accountability for randomized processes must determine why randomness was needed and determine that the source of that randomness meets those goals.

The most intuitive benefit of randomness in a decision policy is that it helps prevent strategic behavior—“gaming” of a system. When a tax examiner, for example, uses software to choose who is audited, randomization makes it impossible for a taxpayer to be sure whether or not he or she will be audited. Those who are evading taxes, in particular, face a partly unknown risk of detection and do not know whether, or when, they should prepare to be audited.

The card game of poker illustrates a second benefit: randomness can obscure secret information. A good poker player has secret information—how good her cards are—that affects how she will bet. By occasionally bluffing, she randomizes her behavior and makes it more difficult for opponents to infer the quality of her hand.

Finally, randomization can give computers more flexibility to perform in unexpected environments. Consider how the Roomba robot is programmed to vacuum rooms. If rules of motion were hard-coded in the software controlling the robot, an unusual furniture configuration might lead to the Roomba getting stuck in a corner or under a table or repeatedly following the same path without cleaning the rest of the room. Adding in randomized decision making strategies are provably more efficient than non-randomized ones. See for example J. C. Gittins (1979). “Bandit Processes and Dynamic Allocation Indices”. Journal of the Royal Statistical Society. Series B (Methodological) 41 (2): 148–177 Even outside machine learning, there are strong indications in computer science theory that certain problems can only be solved efficiently via randomized techniques. Although it is obvious that every efficient algorithm also has an efficient randomized version (which is just rewritten to take some random bits as input and ignore them), it is conjectured but not known that the converse is not true, namely that every efficient randomized algorithm also has a deterministic version that solves the same problem with comparable efficiency.

\textsuperscript{23} For example, a winning lottery ticket with numbers “1 2 3 4 5” is just as likely to be correct as any other ticket, and yet seems strikingly unlikely. In a similar way, it will always be necessary when randomness is involved in a process to ensure that even possible outcomes that are “correct” in the sense that the system could have produced them are also correct in the sense that they fulfill the goals which necessitated randomness in the first place (e.g., in a lottery, that the winning ticket numbers not be known in advance of their selection and not be influenced by the lottery operators).
motion allows it to escape these patterns and work more effectively, without the need to code in all possible room configurations. Randomized strategies can avoid worst-case outcomes with high probability, no matter how unfriendly the environment turns out to be.24

II. DESIGNING COMPUTER SYSTEMS FOR PROCEDURAL REGULARITY

The first goal in any plan to govern automated decision-making should be to enable the people overseeing the process—whether they are government officials, corporate executives, or members of the public—to know how a computer system makes decisions (or, at the very least, that it makes decisions based on some rule, even if that rule is not fully disclosed). A baseline requirement in most contexts is procedural regularity: each participant will know that the same procedure was applied to her and that the procedure was not designed in a way that disadvantages her specifically.25 This baseline requirement draws on the 14th Amendment principle of procedural due process. Ever since a seminal 19th Century case, the Supreme Court has articulated that procedural fairness requires rules to be generally applicable and not designed for individual cases.26 Similarly, federal legislation articulates the requirement for procedural regularity in administrative agency actions.27 These principles are also enshrined in the Federal Rules of Civil Procedure28 for civil litigation.

24 More concretely, one study showed that computer-generated teaching plans customized to particular students can be less effective than lesson plans without customization if the software model used to tailor lessons to individual performance are trained on large groups that do not capture individual-specific patterns. This failure of “big data” methods trained on large groups of students to properly capture the quirks of a “small data” situation (such as a classroom-sized group of students) can be avoided successfully by adding random deviations from the model’s prediction and tracking the results of these deviations. [CITE].

25 For example, a tax auditing risk assessment should not single out individuals either by name or by identifying characteristics. If a process added extra weight to filers of a particular postal code, gender, and birth month, this could be enough to single out individuals in many cases. See e.g., Paul Ohm, Broken Promises of Anonymity: Responding to the Surprising Failure of Anonymization, 57 UCLA L. Rev. 1701, 1716-27 (2010)(showing that an individual’s identity may be reverse engineered from a small number of data points).


27 See Administrative Procedures Act, 5 U.S.C. §§ 500-596

In this Part, we will demonstrate that tools from computer science can guarantee important elements of procedural regularity when they are incorporated in the initial design of computer systems. Specifically, these tools can assure that:

- The same policy or rule was used to render each decision.
- The decision policy was fully specified (and this choice of policy was recorded reliably) before the particulars of decision subjects were known, reducing the ability to design the process to disadvantage a particular individual.
- Each decision is reproducible from the specified decision policy and the inputs for that decision.
- If a decision requires any randomly chosen inputs, those inputs are beyond the control of any interested party.

After describing these properties and showing how they are achieved, we will apply them to a case study—the Diversity Visa Lottery at the State Department—where application of these tools could have greatly improved the legitimacy and fairness of an automated decision procedure.

A. Transparency and its Limits

A naïve solution to the problem of verifying procedural regularity is to demand transparency of the source code, inputs and outputs for the relevant decisions: if all of these elements are public, it seems easy to determine whether procedural regularity was satisfied. Indeed, full or partial transparency can be a helpful tool for governance in many cases, and transparency has often been suggested as a remedy to accountability issues for computerized systems. However, transparency alone is not sufficient to provide accountability in all cases.

First and foremost, it is often necessary to keep secret elements of a decision policy, the computer systems that implement it, key inputs, or the outcome. Keeping aspects of a decision policy secret can help prevent strategic behavior—“gaming” of a system. For example, the IRS may look for telltale signs in tax returns that are highly correlated with tax evasion based on returns previously

29 See Danielle Citron & Frank Pasquale, The Scored Society: Due Process for Automated Predictions, 89 Wash. L. Rev. 1, 8 (2014) (“transparency of scoring systems is essential”)
30 See 14 C.F.R. Part 255.4 (requiring transparency for reservation system display information); Frank Pasquale, Beyond Innovation and Competition: the Need for Qualified Transparency in Internet Intermediaries, 104 Northwestern L. Rev. 105 (2010).
audited. But if the public knows exactly which things on a tax return are treated as telltale signs of fraud, tax cheats may adjust their behavior and the signs may lose their predictive value for the agency. Moreover, when the decision being regulated is a commercial one, such as an offer of credit, a business’s legitimate interest in protecting proprietary information or guarding trade secrets may be incompatible with full transparency. And in many contexts, an automated decision may use as inputs, or will create as an output, sensitive or private data that should not be broadly shared to protect business interests, privacy, or the integrity of law enforcement or investigative methods. Secrecy discourages strategic behavior by participants in the system and prevents violations of legal restrictions on disclosure of data.

Second, while transparency allows for the static and dynamic analyses described in Section I.A, those methods are often insufficient to verify properties of software systems, if these systems have not been designed with future evaluation and accountability in mind.

Third, for decision processes that involve some element of randomness, even full transparency – of the system’s source code, its inputs, and its results – does not guarantee that an outcome cannot be improperly fixed in an undetectable way, as was described in Section I.B. A decision maker could even construct a plausible but misrepresentative audit trail: for example, the decision maker could run the software implementing a random choice multiple times until obtaining a desired outcome, and then keep the audit trail corresponding to the hand-picked result. A simple lottery provides an excellent example: a perfectly transparent algorithm—use a random number generator to assign a number to each participant and have the participants with the lowest numbers win—yields results that cannot be reproduced or verified because the random number generator will produce new random numbers when it is called upon later.

Fourth and finally, systems that change over time cannot be fully understood through transparency alone. System designers regularly change complicated automated decision processes—such as search engine ranking methodology, spam filter rules, intrusion detection system methods, or the algorithms that select website ads—in response to strategic behavior by participants in the

---

31 There are ways to incorporate randomness that can be replicated, which are discussed below in Section II.B.3.
system. Computer systems that choose social media posts to display to users might respond to user behavior. “Online” machine learning systems can update their model for predictions after each decision, incorporating each new observation as part of their training data. Even knowing the source code and data for such systems is not enough to replicate or predict their behavior—we also must know precisely how and when they interacted or will interact with their environment.

B. Auditing and its Limits

One approach to the assurance of procedural regularity is to combine transparency with auditing. Auditing treats a decision process as a black box, of which one can see only the inputs and outputs, but not the inner workings.32 The approach has a long history in offline contexts, such as testing for discrimination in retail car negotiations.33 For retail car negotiations, the transparency of the bargaining process for the purchase of a care is insufficient to determine if different prices are offered based on race or gender.34

Computer scientists, however, have shown that black-box evaluation of systems is the least powerful of a set of available methods for understanding and verifying their behavior.35 Even for measuring demonstrable properties of software systems, such as testing whether a system functions as desired without bugs, it is much more powerful to be able to understand the design of that system and test it in smaller, simpler pieces than to attempt to review its failures simply by looking at how the output responds to changes in input.36 Instead, software developers regularly use other, more powerful evaluation techniques such as white-box testing (in which tests are designed to take advantage of the internal structure of the software code to maximize their ability to

33 See Ian Ayres, Fair Driving: Gender and Race Discrimination in Retail Car Negotiations, 104 Harvard L. Rev. 817 (1991)
34 Id.
35 [CITE]
36 For example, if the output of a system is an error or other failure such as a crash, it is not obvious to an analyst how to modify the output to learn much at all.
find bugs) or the use of programming language features such as types to ensure that their programs perform as expected.\textsuperscript{37}

\textbf{C. Technical Tools for Procedural Regularity}

As we demonstrated above, transparency and auditing often do not suffice for accountability. In this section, we introduce computational methods that can provide accountability for procedural regularity even when some information is kept secret. These methods can be used alongside transparency and auditing when appropriate.

Our approach harnesses the power of computational methods and does not take the design of the computer system as given—indeed, we explicitly advocate for systems to be designed to use methods such as the ones described here. Nor do we give up on governance when all or part of a computer system must remain secret. We rely on several advanced techniques from computer science to enable the governance of secret decision systems: cryptographic commitments, zero-knowledge proofs, and fair random choices. Counterintuitively, even when a piece of software or the data input to it are secret, these methods can guarantee that the software and inputs satisfy the requirements for procedural regularity; namely that the same decision policy was used for each decision; that that policy was determined and recorded before inputs were known; and that outcomes are reproducible. Just because a given computer system or piece of software is secret does not mean that nothing \emph{about} that system can be known.

\textbf{1. Cryptographic Commitments}

Cryptographic commitments are the digital equivalent of a sealed document held by a third party or in a safe place. It is possible to compute a commitment for any digital object (e.g., a file, a document, the contents of a search engine’s index at a particular time, or any string of bytes). Commitments are a kind of promise that \textit{bind} the committer to a specific value for the object being committed to (i.e., the object inside the envelope) such that the object can later be revealed and anyone can verify that the commitment corresponds to that digital object. In this way, as in the envelope analogy, an observer can be certain that the object was not change since the commitment was issued and that the committer did indeed know the value of the object at the time the commitment was made (e.g., the source code to a program or the

\textsuperscript{37} See supra., note 16.
contents of a document). Importantly, secure cryptographic commitments are also *hiding*, meaning that knowledge of the commitment (or possession of the envelope in the analogy) does not confer information about the contents. In this way, the commitment *binds* the committer to the committed-to digital object. Thus the sealed document analogy: once an object is “inside” the sealed envelope, an observer cannot see it nor can anyone change it. However, unlike physical envelopes, commitments can be published, transmitted, copied, and shared at very low cost and do not need to be guarded to prevent tampering. In practice, cryptographic commitments are much smaller than the digital objects they represent.38 Because of this, commitments can be used to lock in knowledge of a secret (say, an undisclosed decision policy) at a certain time (say, by publishing them or sending them to an oversight body) without revealing the contents of the secret, while still allowing the secret to be disclosed later (e.g., in a court case under a discovery order) and guaranteeing that the secret was not changed in the interim (say, the decision policy was modified from one that was explicitly discriminatory to one that was neutral).39

When a commitment is computed from a digital object, the commitment also yields an *opening key*, which can be used to verify the commitment. Importantly, a commitment can only be verified using the precise digital object and opening key related to its computation: it is computationally implausible for anyone to discover either another digital object or another opening key which will allow the commitment to verify properly. In the envelope metaphor, this is tantamount to proof that neither the envelope nor

---

39 As a curiosity, we remark that the popular board game *Diplomacy* is essentially based on physical world commitments: each player negotiates a set of moves for the next round of the game, but then these moves are written on paper and passed secretly to a game master who stores them in an envelope. Once all players have entered their moves, the moves are revealed and taken simultaneously. This commitment mechanism allows players to simulate simultaneous moves without any risk that a player will fall behind or change their moves in a particular round in response to their perception of what another player is doing in that round. However, the commitment mechanism alone does not prevent players from entering incorrect or impossible moves, simply writing nonsense on their paper instead of moves, or simply refusing to enter a move at all (the game master, however, enforces that all moves entered into the envelope are correct and all the players must trust her to do this to ensure that the game is not spoiled). Below, in the section on zero knowledge proofs, we describe how techniques from computer science can address the role of the game master purely through computation.
the document inside the envelope was replaced clandestinely with a different envelope or document. Any digital object (e.g., a file or any string of bytes) can have a commitment and an opening key such that it is: 1) impossible to deduce the original object from the commitment alone; 2) possible to verify, given the opening key, that the original object corresponds to the commitment, and 3) impossible to generate a fake object and fake opening key such that using the (real) commitment and the fake opening key will reveal the fake object.

Cryptographic commitments can be used to ensure that the same decision policy was used for each of many decisions, and that rules implemented in software were fully determined at a specific moment in time. A government agency or other organization that wants to prove that it was using a particular decision policy, that it used particular data as input to a decision policy, or that it computed a particular outcome from that policy and input data could commit to these assertions by taking either its secret source code or the private input data or computed outcome and computing a commitment and opening key (or a separate commitment and opening key for each one). The agency would publish the commitment or commitments publicly and in a way that establishes a reliable publication date, perhaps in a venue such as the Federal Register. Later, the agency could prove that it had the source code, input data, or computed results at the time of commitment by revealing the source code and the opening key to an oversight body such as a court. This technique assures that the software implementing the decision policy was determined and recorded prior to the publication of the commitment, which can be useful to demonstrate that neither the software nor the decision policy were influenced by later information or events.

By themselves, however, cryptographic commitments do not prevent the committer from lying and generating a fake commitment that it cannot open at all or from destroying (or refusing to disclose) the information that allows a valid commitment to be opened. In either case, when the time comes to reveal the contents of the commitment, it will be demonstrable that the committer has misbehaved. However, an observer does not know the nature of the misbehavior: the committer may not have a correct opening key (analogous to having sealed a gibberish or irrelevant document in a physical envelope) or may want to lie about what was in the original file (analogous to discovering that the contents of the envelope may be embarrassing under scrutiny of oversight). In either case, an oversight authority might punish the committer for lying and assume the worst about the contents of
the missing file. However, it would be preferable to be able to avoid this scenario altogether, which we can do with another tool, zero-knowledge proofs, described below.

2. Zero-Knowledge Proofs

A zero-knowledge proof is a two-player game in which one player convinces the other that she knows a secret without revealing what the secret is. As an example, consider the following contrived scenario: Alice doesn’t believe that Bob knows the password to his laptop, so Bob shows the locked laptop to Alice, and then flips it around so Alice cannot see the keyboard or screen and unlocks it, flipping it back to demonstrate that it is no longer locked. In this way, Bob has convinced Alice that he knows something she did not previously believe he did (namely, that he knows his laptop password). However, he accomplished this without actually disclosing the secret password to her.

Counterintuitively, recent advances in computer science show us how to build zero-knowledge proofs that are non-interactive: games such as the one above that can be played in one round, where the first player simply sends a single message to the second player. In fact, we can construct an appropriate non-interactive zero-knowledge protocol to prove the knowledge of nearly anything. These proofs may be structured so that anyone can verify their correctness or alternatively so that only certain pre-designated people can do so.

In the context of cryptographic commitments, zero-knowledge proofs allow the committer to prove that the hidden value inside a commitment has a certain property, but without having to exhibit that value directly or explain how that property is known. For example, if a decision maker commits separately to a specific policy, the inputs to a particular decision based on the policy, and the application of the policy to those inputs for an outcome, a non-interactive zero-knowledge proof can prove that these values correspond to each other, namely that the outcome of applying the committed-to policy to the committed-to input data is the committed-to outcome. This allows decision makers to build audit logs which can be verified by the public to confirm that the decision maker applied the appropriate input to a particular policy.

40 A parallel to this assumption is a spoliation inference, which sanctions a party who withholds, tampers, or destroys evidence by assuming that the missing or changed evidence was unfavorable to the spoliator. See Fed. R. Civ. Pro. §37(e)(2)(A).
in order to reach the stated outcome, all without revealing the
decision policy or private data in the input or outcome. Later, if the
outcome is challenged, a court or other oversight body can compel
the decision maker to reveal the actual policy and input used and
can verify that this matches the published commitments,
effectively providing digital evidence that the decision maker was
honest about its announced decision. By using a commitment to
the same policy in decisions for multiple decision subjects, a
decision maker can demonstrate that they are applying a consistent
policy across the board.

Such zero-knowledge proofs can be enhanced to test parts of
the decision policy, either by exhibiting properties of the
input/output relation (e.g., that a credit score would have been the
same if the subject’s gender were reversed) or properties of the
policy itself (e.g., that the policy only uses certain inputs for
certain purposes.) In Part III, we consider recent work in computer
science on designing machine learning systems that avoid
discrimination.

3. Fair Random Choices

Where random choices are part of a decision-making process,
the fairness of the randomness used in those computer systems
should be verifiable. As briefly indicated in Section II.A, poorly
designed randomization can lead to unaccountable automated
decisions. The decision-maker could influence the supposedly
random choices or could generate many sets of random values and
then pick the set that gives its preferred outcome. Additionally, a
randomized process is not easily reproduced. For example, if it
depends on interaction with its environment (e.g., the operating
system on which it is running, its human user), its behavior may be
altered in a nondeterministic way since that environment can
change between runs.\footnote{One specific example is a program that chooses a random value based on the
time that it has been running but that takes different amounts of time to run
based on what other programs are running on the same physical computer
system.}

Automated decision making processes must be designed from
the beginning to allow for oversight of the decision-maker and to
avoid problems with unpredictable behavior. A decision-maker
could demonstrate that any unpredictable behavior or random
choices in the software do not affect the eventual output: for
example, a program designed to find the top of a hill can start at
any randomly chosen point and take any arbitrary path upwards and will still ultimately return the same maximum value.

More often, the random choices made by an automated decision process will affect the results. In these cases, the software implementing the decision can always be redesigned to replace the set of random choices made by the software with a small recorded, random input (a seed value) from which any necessary random values can be computed in a deterministic, pseudorandom way. In this way, the decision-making process can be replayed so long as the seed is known and the randomness of the input reduces to the randomness of the seed. Using this technique, a decision-maker would not have to generate a new random choice each time a random value is needed by a piece of software (such choices can be made by a cryptographic algorithm that uses the seed to yield reproducible values), nor know in advance how many random choices must be made. This technique allows software that makes random choices, such as a lottery, to be made fully reproducible and reviewable.

If this technique is used, we also must prevent the decision maker from tampering with the seed value, as it fully determines all random data accessed by the program implementing the decision policy. Several methods can aid in ensuring the fair choice of seed values. A public procedure can be used to select a random value: for example, rolling dice or picking ping pong balls from the sort of device used by state lotteries. Alternatively, the seed value could be provided by a trusted third party, such as the random “beacon” operated by the National Institute of Standards and Technology. Perhaps the best option is for a set of mutually

42 Currently known strategies for generating public random values (“randomness beacons”) all have advantages and disadvantages. Dice could be weighted; ping pong balls could be put in the freezer and the cold ones picked out of the machine. The National Institute of Standards and Technology runs a randomness beacon that has come under scrutiny because of distrust of the National Security Agency. The algorithm designer should pick the source of randomness most likely to be trusted by participants, which may vary. Or, the algorithm designer could choose to collect many sources of random choices and mix them together to maximize the number of participants who will trust the randomness of the chosen seed.

43 Computer science refers to a trusted third-party source of randomness as a “beacon”. The best known beacon is operated by the U.S. National Institute of Standards and Technology (NIST), which publishes new random data every few minutes, ostensibly based on the measurement of the radioactive decay of a sample of uranium maintained in a NIST lab. See US Dept. of Commerce, NIST Randomness Beacon, http://www.nist.gov/itl/csd/ct/nist_beacon.cfm (Dec. 22, 2015). Recent revelations about NIST’s role in allowing the U.S. National
distrustful parties (possibly including decision subjects themselves) to engage in an interaction out of which comes a value which is unpredictable so long as at least one participant provided random input. The simplest form of this would involve a decision subject entering a short random number as part of the input for their decision (e.g., on an application form). Then, the decision-maker could generate a seed value for each decision by combining this public random value with a private random value to which the decision-maker committed far in advance of seeing the public random value and a unique identifier for the particular decision (e.g., the social security number of the participant). In some cases, it might make sense for this combination step to happen as part of the decision process. In order to foster maximum confidence that random choices are not improperly influenced, decision makers should derive them using a combination of a random value from a trusted third-party; some random value chosen by the decision-maker and possibly kept secret; and a participant or decision-specific identifier that cannot be changed or controlled by the decision-maker, such as a social security or identification number or other immutable piece of the subject’s name or data. Since these values are either outside of the decision-maker’s control or committed to before knowing the inputs, using these methods gives assurance that the decision-maker is not skewing the results by controlling the selection of random values.

Security Agency to undermine the security of random number generation techniques standardized by NIST have led to some distrust of the NIST beacon, although it may be trustworthy in some applications. Other beacon implementations have been proposed, including beacons based on “cryptocurrencies” such as Bitcoin. See Joseph Bonneau, Jeremy Clark, & Steven Goldfeder, On Bitcoin as a public randomness source (October 2015) https://eprint.iacr.org/2015/1015.pdf

Computer science has methods to simulate a trusted third party making a random choice. These methods require the cooperation of many mutually distrustful parties, such that as long as any one party chooses randomly, the overall choice is random. By selecting many participants in this process, one can maximize the number of people who will believe that the chosen value is in fact beyond undue influence.

When the fairness of random choices is key to the accountability of a decision process, great care must be taken in determining the source of random seed values, as many very subtle accountability problems are possible. For example, by changing the order in which decisions are taken, the decision maker can effectively “shop” for desirable random values by computing future deterministic pseudorandom values and picking the order of decisions based on its preference for which decisions receive which random choices. To prevent this, it may also be necessary to require that the decision maker take decisions in a particular order or that the decision maker commit to the order in which it will
D. Applying Technical Tools to Reform the Diversity Visa Lottery

Armed with these tools, we can turn to the question of how to ensure the procedural regularity of automated decision-making. To illustrate how designing a computer system can make it more accountable, we will apply the methods described above to a case study: the Diversity Visa Lottery (DVL) operated by the United States State Department.

1. Current DVL Procedure

The DVL is run annually by the State Department to grant U.S. permanent resident visas ("green cards") to 50,000 immigrants from around the world. The process, prescribed by 8 U.S.C. 1153(c), is intended to increase the national and regional diversity of immigrants to the U.S., by granting visas to a sample of people from countries otherwise underrepresented in the immigrant population.

The annual DVL process operates as follows. Would-be immigrants can apply to be entered in the lottery. Applicants are grouped according to their country of birth. Within each country group, applicants are put into a rank-ordered list in a random order (the lottery step). A number of applicants to accept from each country is calculated, using a formula based on the number of immigrants to the U.S. in recent years from each country and region. The calculated number of applicants is selected from the top of each country’s rank-ordered list. These "winners" are screened for eligibility to enter the U.S., and they receive visas if they are eligible.

Questions have been raised about the correctness and accountability of this process. Would-be immigrants sometimes question whether the process is truly random or, as some suspect, is manipulated in favor of individuals or groups favored by the U.S. government. This suspicion, in turn, may subject DVL winners to reprisals, on the theory that winning the DVL is evidence of having collaborated secretly with U.S. agencies or interests.

There have also been undeniable failures in carrying out the DVL process. For example, the 2012 DVL initially reported
incorrect results, due to programming errors coupled with lax management.\textsuperscript{46}

An accountable implementation of the DVL could address both issues, by demonstrating that there is no favoritism in the process, and by making it easy for outsiders to check that the process was executed correctly.

2. Transparency Is Not Enough

The DVL is an automated decision system for which transparency alone cannot solve its problems. First, the software implementing the decisions appears to be written in an irreproducible way.\textsuperscript{47} The system relies on the computer’s operating system to provide random numbers; thus, attempts to replicate the program’s execution at another time or on another computer will yield different random numbers and therefore a different DVL result. Notably, no amount of reading, analyzing, or testing of the software can remedy the non-replicability of this software.

Second, the privacy interests of participants bars full transparency. People who apply to the DVL do not want their information or even the fact that they applied to be published. However, such publication is needed for the process to be verified through transparency and auditing. One might try to work around this problem by assigning an opaque record ID to each applicant, and then having the lottery choose record IDs rather than applicants, but lottery operators could manipulate the outcome by retroactively assigning winning record IDs to people they wanted to favor.

3. Designing the DVL for Accountability

Instead of this inherently unverifiable approach, we sketch here a technical solution for building an accountable version of the DVL.\textsuperscript{48} Using the techniques we have described, the State Department can demonstrate that they are running a fair lottery among a hidden set of participants.\textsuperscript{49}

\textsuperscript{47} Id.
\textsuperscript{48} A full technical analysis is beyond the scope of this paper.
\textsuperscript{49} Note that it is less straightforward to prove that the set of participants actually considered in the lottery matches the set of individuals who applied to be included. For example, the operator of the lottery might insert “shills”, or lottery
We can solve the non-replicability problem by choosing a random seed as described in Section II.B.3. In this instance, the third-party generating the random value used to create the seed perhaps could be one or more trusted NGOs.

Recall that the decision policy for the DVL is fixed in statute and hence publically known. To allow oversight, the State Department could publish in the Federal Register a commitment to its software source code (far in advance of any decisions being made) and a commitment to all the inputs (i.e. to each data element in an application for the US visa) provided to the code used to create the rank ordered list and calculate the cut-off points. The State Department also should provide zero-knowledge proofs that applying the committed-to software to the committed-to inputs produce the announced lottery results. The proof should also demonstrate that the commitment to the software published in advance of all decisions is a commitment to the same software as the one used in each individual decision. These actions bind the State Department to its choices of software source code and applicant data, ensure that the commitment to the software was not a fake, and prove that the same procedure was used to render each decision. Subsequent auditing by an oversight body can establish that the source code in the commitment implements the policy specified by statute faithfully (the code should be designed to enable this).

Finally, the State Department should determine an adequate method for generating a random seed to be used in the lottery step. This method should guarantee to the public in general that it is not possible for the State Department to choose winners by rearranging applications.\(^5\)\(^0\) This could be accomplished by combining random data chosen via a public ceremony (as is done for state lotteries) or through the cooperation of the State Department with interested NGOs to produce a verifiable random seed with a random value selected by the State Department on its own (and published prior to the ceremony and any lottery applications) along with something entries that do not correspond to any real applicant, and if one of these applications were to be chosen, that place could be given improperly to anyone of the Department’s choosing. It is technically nontrivial to prove that no extra applications were considered; studies of end-to-end secure voting protocols provide methods to do so.

\(^5\)\(^0\) Random choices in the DVL must be demonstrably random even to non-participants so that winners can plausibly claim that they were chosen by lottery and not because of sympathy for U.S. interests.
that identifies a particular lottery entry uniquely (e.g., the applicant’s full application data, reduced by cryptographic hash, to a small numeric value). Depending on the implementation and application, the State Department could perhaps also include randomness selected by DVL applicants on their application (which could be harvested passively by e.g. tracking mouse movements during the application process).  

Once these steps are taken, each applicant can be assured that his decision is fully explainable. If the applicant wants to question the process or a governmental overseer wants to audit it, the decisions will be replicable and, if necessary, the secret source code and secret input data (including the random choices made in the lottery step) can be revealed and verified—by a court or auditing agency—to be the proper code and data used to render the decision.

These solutions depend on both redesigning the software code (a technical solution) and adopting procedures relating to how the software program is used (a legal or policy solution). They also must be deployed during the design of the decision process and cannot salvage a poorly designed system after the fact.

In hindsight, it should not be surprising that the path to accountability for computational processes requires some redesign of the processes themselves. The same was true for non-computational administrative processes, where the most accountable processes are those that are designed with accountability in mind.

III. DESIGNING ALGORITHMS TO ASSURE FIDELITY TO SUBSTANTIVE POLICY CHOICES

In the previous Part, we described methods for ensuring that automated decisions are reached in accordance with agreed rules, a goal we called procedural regularity. We have described how to determine if a program is implemented properly and how to apply oversight to ensure legal compliance. But even when a rule is

51 See supra, note 46.
52 In fact, just as the applicant can be convinced that his decision is explainable without seeing the secret algorithm or secret inputs, an oversight body can be convinced that particular decisions were made correctly without seeing the applicant’s inputs, which might contain sensitive data, such as health records or tax returns. Thus, subsequent auditing is rendered more useful and more acceptable to participants, as it can determine the basis for every decision without needing to examine sensitive information.
properly applied in each case, people may still wonder whether the rule is correctly implemented, moral, legal, ethical, or if it operates in aggregate with fidelity to substantive policy choices. For example, whether a properly applied algorithm unfairly disadvantages legally protected groups may still be unknown—whether, in the policymaker’s sense of the term, the rule does not comport with the policy choice and is discriminatory.

In this Part, we look at tools available to assure that substantive policy choices are effectively implemented in automated decisions beyond the simple determination that rules are consistently applied. Specifically, we address how and where computer science can help to illuminate and address undesirable consequences that even a consistently applied rule may have for well-defined policy choices. To address this capability of computer science, we examine how a computerized decision rule may discriminatory and the potential role of technical tools to address the problem.\footnote{The word “discrimination” carries a very different meaning in engineering conversations than it does in public policy. Among computer scientists, the word is a value-neutral synonym for differentiation or classification: a computer scientist might ask, for example, how often a facial recognition algorithm successfully discriminates between human faces and inanimate objects. But for policymakers, “discrimination” is most often a term of art for invidious, unacceptable distinctions among people – distinctions that either are, or reasonably might be, morally or legally prohibited.}

What makes a rule \textit{count} as unacceptably discriminatory against some person or group is a fundamental, and fundamentally contested, question. We do not address that question here, much less claim to resolve it with computational precision. Instead, we describe how an emerging body of computer science techniques may be used to avoid outcomes that could be considered discriminatory.

Fidelity to policy choices like non-discrimination is a more complicated goal than procedural regularity, and the solutions that currently exist to address it are less comprehensive. Technical tools offer some ways to ameliorate these problems, but they generally require a well-defined notion of what sort of fairness they are supposed to be enforcing. Violations of procedural regularity are clear-cut; violations of principles of fairness are often murky and difficult to define, and thus to demonstrate. In addition, the
precision of computer code often brings into sharp focus the
tensions within current legal frameworks for anti-discrimination.
Computers favor hard and fast rules over standards and balancing
tests found often in our common law system in civil rights law and
policy. This suggests that doctrinal reform may be necessary
before computerized decision-making can be satisfactorily applied
in an area. Here, we offer an overview of the problem of
algorithmic discrimination and the current state of the related
technical tools and their relationship to existing legal frameworks.
Our aim is both to elucidate the current state of the art and to
suggest directions for further research and action.

A. Possible Discriminatory Effects of Algorithms

A significant concern about algorithmic decision-making is
that it may simultaneously systematize and conceal discrimination.
Because it can be difficult to predict the effects of a rule in
advance (especially for large, complicated rules or rules that are
machine-derived from data), regulators and observers may be
unable to tell that a rule has discriminatory effects. In addition,
decisions made by algorithms may enjoy an undeserved aura of
fairness or objectivity.\footnote{Paul Schwartz, Data Processing and Government Administration: The Failure of the American Legal Response to the Computer, 43 Hastings L. J. 1321, 1341 (1992) (describing the deference that individual’s give to computer results as the “seductive precision of output”).} However, the design and implementation
of algorithms is vulnerable to a variety of problems that can result
in systematically faulty and biased determinations.\footnote{Id.}

We focus on decision procedures developed through machine
learning—on situations where a machine has been “trained”
through exposure to a large quantity of data, inferring a rule from
the patterns it observes. Computers are especially well-suited to
discovering patterns in these input-output pairs that can then guide
future decision-making. In contrast to human-made rules, these
rules for decision-making are induced from historical examples—
quite literally rules learned by example. Humans orchestrate a
computerized rule-creation process, rather than imparting the rules
directly.

Machine learning is an increasingly common approach to
solving problems that once seemed computationally intractable due
to their complexity (e.g., object recognition). The recent movement
of algorithms into a growing number of domains owes primarily to
successful applications of machine learning, which is thus the primary focus of our analysis.

These algorithms are machine-made, but the lessons they embody may be biased or unfair nevertheless. We describe below a few illustrative ways that algorithms generated through machine learning may turn out to be discriminatory, adapting a taxonomy laid out in previous work by Solon Barocas and Andrew D. Selbst.  

First, algorithms that include some type of machine learning can lead to discriminatory results if the algorithms are trained on historical examples that reflect past prejudice or implicit bias, or on data that offer a statistically distorted picture of groups comprising the overall population. Tainted training data would be a problem, for example, if a program to select among job applicants is trained on the previous hiring decisions made by humans, and those previous decisions were themselves biased. Statistical distortion, even if free of malice, can produce similarly troubling effects: consider, for example, an algorithm that instructs police to stop and frisk pedestrians. If this algorithm has been trained on a dataset that over-represents the incidence of crime among certain groups (because these groups have historically been the target of disproportionate enforcement), the algorithm may direct police to detain members of these groups at a disproportionately high rate (and non-members at a disproportionately low rate). Such was the case with the New York City Police Department’s stop-and-frisk program, where data from 2004 to 2012 showed that 83% of the stops were of a black or Hispanic person and 10% were of a white person, in a resident population that was 52% black or Hispanic and 33% white. Note that the over-representation of black and Hispanic people in this sample may lead an algorithm to associate

57 Solon Barocas & Andrew Selbst, Big Data’s Disparate Impact 11-12 (citing Stella Lowry & Gordon Macpherson, A Blot on the Profession, 296 Brit. Med. J. 657, 657 (1988)). Another example is the Google algorithm that shows ads for arrest records much more frequently when black-identifying names are searched than when white-identifying names are searched—likely because users click more often on arrest record ads for black-identifying names and the algorithm learns from this behavior with the purpose of maximizing click-throughs. Id. at 12-13 (citing Latanya Sweeney, Discrimination in Online Ad Delivery, 56 Comm. ACM 44 (2013)).
typically black or Hispanic traits with stops that led to crime prevention, simply because those characteristics are over-represented in the population that was stopped.59

Second, machine learning models can build in discrimination through feature selection, meaning through the choice of inputs. Three types of choices about inputs could be of concern: using membership in a protected class directly as an input (e.g., decisions that take gender into account explicitly); considering an insufficiently rich set of factors to assess members of protected class with the same degree of accuracy as non-members (e.g., in a hiring application, if fewer women have been hired previously, data about female employees might be less reliable than data about male employees), and relying on factors that happen to serve as proxies for class membership (e.g., decisions that hinge on place of residence when neighborhoods are highly segregated). The case against using a proxy is clearer when alternative inputs could yield equally effective results with fewer disadvantages to protected class members. As described in Section III.B.2, however, there are cases where allowing an algorithm to consider protected class status can actually make outcomes more fair. This may require a doctrinal shift, as in many cases, consideration of protected status in a decision is presumptively a legal harm.

Third and finally, there is the problem of “masking”: intentional discrimination hidden as one of the above-mentioned forms of unintentional discrimination. A prejudiced decision-maker could skew the training data or pick proxies for protected classes with the intent of generating discriminatory results.60 More pernicious masking could occur at the level of designing a machine-learning model, which is a very human-driven, exploratory process.61

Hardt62 and Dwork et al63 propose a “catalog of discriminatory evils”, where each entry is a form of intentional discrimination that

59 The under-representation of white people would likely cause the opposite effect, though it could be counter-balanced if, say, the police stopped a subset of white people who were significantly more likely to be engaged in criminal behavior.

60 See Solon Barocas & Andrew Selbst, Big Data’s Disparate Impact 22-23.

61 In other words, the machine learning model would be intentionally coded to develop bias.

62 See infra, note 84.

63 See infra, note 86.
is increasingly more difficult to detect. These correspond with—and nicely explain—parts of the above taxonomy.

1. **Blatant, explicit discrimination.** When an algorithm explicitly tests the data of individuals for membership in a protected group and gives different chances of each possible outcome to individuals in the group as compared to those who are not. [This corresponds to feature selection using proscribed features]

2. **Discrimination based on a redundant encoding.** Rather than testing specifically for some protected status, an algorithm can replace that test by a test which is essentially equivalent but based on access only to “allowed” data. Such unprotected attributes that correlate strongly with protected status are often called *proxies* for the protected status. For example, a test for race might be replaced by a very localized test for residential address in a credit decision, or gender might be replaced by a test for specific magazine subscriptions in an employment screening decision. [This corresponds to masking using intentional proxies]

3. **Redlining.** The practice of discriminating by treating a redundant encoding of protected status as if it were the protected status itself, so as to discriminate against anyone else who might share that attribute collaterally. For example, a test for race could be replaced by a test for residential address at the level of neighborhood, ZIP code, or city name, so that even people with the “preferred” value of the protected attribute will be given adverse treatment. [This corresponds to masking using purposefully coarse features]

4. **Cutting off business with or giving adverse decisions to a segment of the population in which membership in a protected status is disproportionately high.** A generalized version of redlining, in which a decision maker targets proxy attributes where members of a protected status group are more likely to be found than in the general population, even if they might not represent the majority of people with that attribute. For example, a decision maker who wants to offer worse financial terms to patients with a certain terminal disease might instead target anyone who regularly

---

64 Barocas and Selbst, *supra*, note 56.
purchases a certain combination of over-the-counter drugs, even if that combination is widely used. [This corresponds to masking using intentional proxies, with the additional strategy of further masking the use a proxy by relying on a proxy that separates out groups composed disproportionately—but not exclusively—of members of a protected class]

5. Self-fulfilling prophecy. By deliberately choosing the “wrong” members of a protected status group, a decision maker can build a poor track record for that group or design a system which appears to be fair on cursory inspection but in fact has significantly disparate effects in practice. For example, an advertiser can show an ad for a high-end resort only to members of the protected status group it knows cannot afford to go, while showing the same ad to qualified nonmembers of the protected group. The end effect will be that the protected group is excluded, even if the ad was shown equally to protected status individuals and non-protected status individuals. A more subtle version would be to show protected status individuals an ad which is unlikely to generate a response while showing non-protected status individuals ads which are likely to generate conversions. In either case, the solution is to treat similar individuals similarly, without regard to their protected attribute status, so that qualified members of both groups are shown similar advertisements. [This corresponds to masking using purposefully skewed samples]

6. Reverse tokenism. Imagined in the context of finding a convincing refutation of claims of discrimination based on protected status, a decision maker can adopt this practice of making an adverse decision about one or a few non-protected status individuals, so as to exhibit them later as token rejectees in order to provide evidence that protected status was not the dispositive factor in a decision. Again, this problem is solved by the general practice of requiring that similar individuals be treated in similar ways, such that it would be impossible for a decision maker to treat some obviously well-qualified individual in an adverse manner without significantly hobbling their ability to make broadly accurate decisions. [This corresponds to masking using purposefully tainted examples, though, in this case, the decision maker has purposefully discounted a qualified candidate from the unprotected group]
As we will show, even harms of this sort lend themselves to technical mitigations. Further, attempts to address the varieties of intentional discrimination described above will also provide useful solutions to their unintentional equivalents. Computer scientists, including Hardt\textsuperscript{65} and Dwork et al\textsuperscript{66}, have developed techniques that formalize fairness in such a way that it can then constrain the machine learning process. We hope not only to equip designers with these tools, but to convince observers and users of such systems that those systems have been designed in such a way.

\textbf{B. Technical Tools for Non-Discrimination}

As mentioned in the previous Part, transparency and after-the-fact auditing can only go so far in preventing undesired results. Ideally, those types of ex post analysis should be used in tandem with powerful ex ante techniques during the design of the algorithm.

Several ideas from computer science can help with the design of processes that use algorithms for decision-making to ensure that those processes minimize discrimination (and can be certified as such by designers, participants, or overseers): 1) the incorporation of randomness into decision outcomes to maximize the gain of learning from experience; 2) approaches to maximize fairness in machine learning systems; 3) related ideas from differential privacy; and 4) the zero-knowledge proofs of section II.B.2.

1. Learning from Experience

As mentioned in Section I.B, incorporating randomness into an algorithm can give it flexibility to operate outside of the environment for which it was designed. Similarly, randomness can prevent hidden biases in the design or deployment of an algorithm from leading to consistent discriminatory outcomes. There is a large and rich literature on how to maximally learn from previous data and how to use random choices to ensure that a model is as faithful as possible to the real world.\textsuperscript{67}

Consider a machine-learning algorithm for hiring that is trained using a biased set of initial data indicating that women are weak candidates, even though gender does not predict job performance

\textsuperscript{65} See \textit{infra}, note 84.

\textsuperscript{66} See \textit{infra}, note 86.

\textsuperscript{67} This literature is divided between the machine learning research community in Computer Science and the study of optimal decision making in Statistics. See \textit{supra}, note 17.
among the full population. If the resulting model would hire mostly men, the algorithm for hiring can create a self-fulfilling prophecy in which it finds that characteristics of successful hires correlate strongly with proxies for gender. But if the algorithm is designed to incorporate an element of randomness such that some candidates who are not predicted to do well get hired (and have their performance tracked), the validity of the initial assumptions can be tested and the accuracy and fairness of the entire system will benefit over time. By occasionally guessing about candidates for which the model cannot make confident predictions, the model can gather additional data and evolve to become more faithful to the real world.

Similarly, randomness is often necessary when training machine-learning models. Models may become too specialized or specific to the data used for training, a problem called “overfitting.” Randomness can prevent this problem. Likewise, models may find a decision rule well-suited for some portion of the input, but which is not the best rule overall. Randomness can also help avoid this bias. Consider for example a credit-scoring model trained initially on a biased set of data that under-rates the creditworthiness of some minority group. Even if the model is the best possible decision rule for a population matching the biased input data, the model may unfairly deny access to credit to members of that minority group. In addition to the discrimination, the use of this model denies creditors business opportunities with the unfairly rejected individuals. Here again, allowing the model to occasionally guess randomly, combined with tracking of expected versus actual performance, can improve the model’s faithfulness to the population on which it is actually used, rather than the biased population on which it was trained. The information from this injection of randomness can be fed back to the model to improve the accuracy and fairness of the system overall.

2. Fair Machine Learning

One commonly understood way to demonstrate that a decision process is independent from sensitive attributes is to preclude the use of those sensitive attributes from consideration. For example, race, gender, and income may be excluded from a decision-making process to assert that the process is “race-blind”, “gender-blind”, or “income-blind.” 68 From a technical perspective, however, this

---

68 See e.g 12 C.F.R. Part 1002.5(b) (“A creditor shall not inquire about the race, color, religion, national origin, or sex of an applicant or any other person in
approach is naive. Blindness to a sensitive attribute has long been recognized as an insufficient approach to making a process fair. The excluded or “protected” attributes can often be implicit in other non-excluded attributes. For example, when race is excluded as a valid criterion for a credit decision, redlining may occur when a zip code is used as proxy that closely aligns with race.\(^6^9\)

It seems clear that this type of input “blindness” is insufficient to assure fairness and compliance with substantive policy choices. Although there are many conceptions of what fairness means, we consider here a definition of fairness in which similarly situated people are given similar treatment—that is, a fair process will give similar participants a similar probability of receiving each possible outcome. This is the core principle of a developing literature on *fair classification in machine learning*, an area first formalized by Dwork, Hardt, Pitassi, Reingold, and Zemel.\(^7^0\) This work stems from a longer line of research on mechanisms for data privacy. We describe further the relationship between fairness in the use of data and privacy in Section III.B.3.

The principle that similar people should be treated similarly is often called *individual fairness* and it is distinct from *group fairness*, in the sense that a process can be fair for individuals without being fair for groups.\(^7^1\) Although it is almost certainly the more policy-salient, group fairness is more difficult to define and achieve. The most commonly studied notion of group fairness is *statistical parity*, which says that an equal fraction of each group should receive each possible outcome. While statistical parity seems like a desirable policy because it eliminates redundant or proxy encodings of sensitive attributes, it is an imperfect notion of fairness. For example, statistical parity says nothing about whether a process addresses the “right” subset of a group. Imagine an


\(^{7^1}\) Sometimes, a more restrictive notion of individual fairness implies group fairness. *Id.* Intuitively, this is because if people who are sufficiently similar are treated sufficiently similarly, there is no way to construct a minority of people who are treated in a systematically different way.
advertisement for an expensive resort: we would not expect that showing the advertisement to the same number of people in each income bracket would lead to the same number of people clicking on the ad or buying the associated product. As explained in III.A, a malicious advertiser wishing to exclude a minority group from the resort could design its advertising program to maximize the likelihood of conversion for the desired group while minimizing the likelihood that the ad will result in a sale to the disfavored group. In the same vein, if a company aimed to improve the diversity of its staff by offering the same proportion of interviews to candidates with minority backgrounds as are minority candidate applications, that is no guarantee that the number of people hired will reflect the population either of applicants or in general. And the company could hide discriminatory practices by inviting only unqualified members of the minority group, effectively creating a self-fulfilling prophecy as described in Section III.B.1.

The work of Dwork et al. identifies an additional interesting problem with the “fairness through blindness” approach: by remaining blind to sensitive attributes, a classification rule can in fact select exactly the opposite of what is intended. Consider for example a system that classifies profiles in a social network as representing either real or fake people based on the uniqueness of their name. In European cultures, from which a majority of the profiles come, names are built by making choices from a relatively small set of possible first and last names, so a name which is unique across this population might be suspected to be fake. However, other cultures (especially Native American cultures) value unique names, and so it is common for people in these cultures to have names which are not shared with anyone else. However, because a majority of accounts come from the majority of the population, for which unique names are rare, any classification based on the uniqueness of names will inherently classify real minority profiles as fake at a higher rate than majority profiles, and will correspondingly misidentify fake profiles using names drawn from the minority population as real. This unfairness could be remedied if the system were “aware” of the minority status of a name under consideration, since then the algorithm could know whether the implication of a unique name is that a profile is very likely to be fake or very likely to be real.

This insight explains why the approach taken by Dwork et al. is to enforce similar probabilities of each possible outcome on similar people, requiring that the aggregate difference in probability of any individual receiving any particular outcome be limited.
Specifically, Dwork et al. require that this difference in chance of outcome be less than the difference between individuals subject to classification. This requires: (1) a mathematically precise notion of how “different” people are, which might be a score of some kind or might naturally arise from the data in question (for example, if the physical location of subjects is a factor in classification, we might naturally use the distance between subjects as one measure of their similarity); and (2) that this notion of similarity capture all relevant features, including possibly sensitive or protective attributes such as minority status, gender, or medical history. Because this approach requires the collection and explicit use of sensitive attributes, the work describes its definition of fairness as “fairness through awareness”. While the work of Dwork et al. provides only a theoretical framework for building fair classifiers, others have used it to build practical systems that perform almost as well as classifiers that are not modified for fairness.

The work of Dwork et al. also provides the theoretical basis for a notion of fair affirmative action, the idea that imposing an external constraint on the number of people from particular subgroups who are given particular classifications should have a minimal impact on the principle that similar people are treated similarly. This provides a technique for forcing a fairness requirement such as statistical parity even when it will not arise naturally from some classifier.

A more direct approach to making a machine learning process fair is to modify or select the input data in such a way that the output decisions satisfy some fairness property. For example, in order to make sure that a classifier does not over-reflect the minority status of some group, we could select extra training samples from that group or duplicate samples we already have. In either case, care must be taken to avoid biasing the training process in some other way or overfitting the model to the non-representative data.

Other work focuses on fair representations of data sets. For example, we can take data points and assign them to clusters, or groups of close-together points, treating each cluster as a prototypical example of some portion of the original data set. This is the approach taken by Zemel, Wu, Swersky, Pitassi, and

---

72 For technical reasons, this particular formulation is mathematically convenient, although different bounds might also be useful.
Dwork. Specifically, Zemel et al. show how to generate such prototypical representations automatically and in a way that guarantees statistical parity for any subgroup in the original data. In particular, the probability that any person in the protected group is mapped to any particular prototype is equal to the probability that any person not from the protected group is mapped to the same prototype. Therefore, classification procedures which have access only to the prototypes must necessarily not discriminate, since they cannot tell whether the prototype primarily represents protected or unprotected individuals. Zemel et al. test their model on many realistic data sets, including the Heritage Health Prize data set, and determine that it performs nearly as well as best-of-breed competing methods while ensuring substantial levels of fairness. This technique allows for a kind of “fair data disclosure”, in which disclosing only the prototypes allows any sort of analysis, fair or unfair, to be run on the data set to generate fair results.

A related approach is to use a technique from machine learning called regularization, which involves introducing new information to make trained models more generalizeable (usually in the form of a penalty assigned to undesirable model attributes or behaviors). This approach has also led to many useful modifications to standard tools in the machine learning repertoire, yielding effective and efficient fair classifiers.

The approach of Zemel et al. suggests a related approach, which is also used in practice: the approach of generating fair synthetic data. Given any set of data, we can generate new data such that no classifier can tell whether a randomly chosen input was drawn from the real data or the fake data. And further, we can use approaches like that of Zemel et al. to ensure that the new data are at once representative of the original data and also fair for individuals or subgroups. Because synthetic data is generated at random, it is useful when training a classifier on real data would create privacy concerns. Also, synthetic data can be made public for others to use, although care must be taken to avoid allowing others to infer facts about the underlying real data. Such model

---


inversion attacks\textsuperscript{75} have been demonstrated in practice, along with other inference or deanonymization attacks that allow sophisticated conclusions without direct access to the data those conclusions are about.\textsuperscript{76}

All of these approaches demonstrate that it is possible to build a wide variety of definitions of fairness into a wide variety of data analysis and classification systems, at least to the extent that a definition of fairness is known or can be approximated in advance. While no bright-line rules exist that would allow the designer or operator of such a system to guarantee that their behavior is compliant with anti-discrimination law, it is certainly not the case that no options exist or that unconstrained use of data analysis is necessary or even defensible.

Many of these approaches rely on the insufficient notion of group fairness by statistical parity. To the extent that more technical research can help to address the problem of unfairness in big data analysis, it is by expanding the repertoire of definitions of group fairness that can be usefully applied in practice and by providing better exploratory and explanatory tools for comparing different notions of fairness. From a law and policy perspective, it would be extremely useful to system designers to have a set of rules, standards, or best practices that explain what notions of fairness are best used in specific real-world applications.

A complementary notion to machine learning systems that can guarantee pre-specified formal fairness properties is the work of Rudin on machine learning systems that are interpretable.\textsuperscript{77} Such systems generate models that can be used to classify individuals, but also explanations for why those classifications were made. These explanations can be reviewed later to understand why the model behaves a certain way, and in some cases how changes in the input data would affect the model’s decision. These


\textsuperscript{77} Rudin, C. “Algorithms for Interpretable Machine Learning”, 20\textsuperscript{th} ACM Conference on Knowledge Discovery and Data Mining, New York, NY, 24-27 August 2014.
explanations can be extremely valuable to experts and oversight authorities, who wish to avoid treating models as black boxes.

3. Discrimination, Data Use, and Privacy

A different way to define whether a classification is fair is to say that we cannot tell from the outcome whether the subject was a member of a protected group or not. That is, if an individual’s outcome does not allow us to predict that individual’s attributes any better than we could by guessing them with no information, we can say that outcome was assigned fairly. To see why this is so, observe the contrary: if the fact that an individual was denied a loan from a particular bank tells you that this individual is more likely to live in a certain neighborhood, this implies that you hold a strong belief that the bank denies credit to residents of this neighborhood and hence a strong belief that the bank makes decisions based on factors other than the objective credit risk presented by applicants.

Thus, fairness can be seen as a form of information hiding requirement similar to privacy. If we accept that a fair decision does not allow us to infer the attributes of a decision subject, we are forced to conclude that fairness is protecting the privacy of those attributes.

Indeed, it is often the case that people are more concerned that their information is used to make some decision or classify them in some way than they are that the information is known or shared. This concern relates to the famous conception of privacy as “the right of the individual to be let alone” in that generally people are concerned with the idea that disclosure interrupts their enjoyment of an “inviolable personality”.

Data use concerns also surface in the seminal work of Solove, who refers to concerns about “exclusion” in “information processing”, or the lack of disclosure to and control by the subject of data processing and “distortion” of a subject's reputation by way of “information dissemination.” Solove argues these problems can be countered by giving subjects knowledge of and control over their own data. In this framework, the predictive models of

79 Id.
81 Id., at 546.
automated systems, which might use seemingly innocuous or natural behaviors as inputs, create anxiety on the part of data subjects.

We can draw an analogy between data analysis and classification problems and the more familiar data aggregation and querying problems which are much discussed in the privacy literature if we consider decisions about an individual as representing (potentially private) information about that individual. In this analogy, a vendor or agency using a model to draw automated decisions wants those decisions to be as accurate as possible, corresponding to the idea in privacy that it is the goal of a data analyst to build as complete and accurate a picture of the data subject as is feasible.

A naive approach to making a data set private is to delete “personally identifying information” from the data set. This is analogous to the current practice of making data analysis fair by removing protected attributes from the input data. However, both approaches fail to provide their promised protections. The failure in fairness is perhaps less surprising than it is in privacy - discrimination law has known for decades about the problem of proxy encodings of protected attributes and their use for making inferences about protected status that may lead to adverse, discriminatory effects.

The work of Hardt\textsuperscript{84} relates the work on fairness by Dwork et al.\textsuperscript{85} to the work on differential privacy by Dwork.\textsuperscript{86} As differential

\textsuperscript{82} Reidentification of individuals based on inferences from disparate data sets is a growing and important concern that has spawned a large literature in both Computer Science and Law. For an overview, see Ohm, P. “Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization”, 57 UCLA L. Rev. 1701 as well as supra, note 36.

\textsuperscript{83} For example, the law explicitly forbids the (sole) use of certain attributes which are likely to be highly correlated with protected status categories, as in protections against redlining. See e.g 12 C.F.R. Part 1002.5(b)(“ A creditor shall not inquire about the race, color, religion, national origin, or sex of an applicant or any other person in connection with a credit transaction”); 12 C.F.R. Part 1002.6(b)(9)(“ a creditor shall not consider race, color, religion, national origin, or sex (or an applicant's or other person's decision not to provide the information) in any aspect of a credit transaction.”)


\textsuperscript{85} supra, note 31.

privacy is a well-founded notion of protection against inferences and the recovery of an individual identity from “anonymous” data, so is fairness through awareness a sound notion of fairness for individuals and a theoretical framework on which to ground more complicated notions of fairness for protected groups.

The many techniques of building fair data analysis and classification systems described in Section III.B.2 mostly require decision makers to have access to protected status information, at least during the design phase of the algorithm. However, in many cases, concerns about misuse, reuse, or abuse of this information has led to a policy regime where decision makers are explicitly barred from using such information. The deployment of these technical tools would require a policy change.\(^{87}\) The techniques described in Section III.B.4 could be used to make such a change less prone to engendering the very real concerns of data abuses that have led to the current regime.

4. Zero-knowledge Proofs

The zero-knowledge proofs explained in Section II.B.2 can be used to certify properties of an algorithmic decision-making process to observers, participants, or overseers. When an algorithm or its inputs are kept secret, a decision-maker can give a zero-knowledge proof that the algorithm does not have undesirable properties or has desirable ones. For example, one could show that a particular algorithm does not use sensitive or prohibited classes of information as input or that a decision would have been the same for different values of sensitive attributes (e.g. that a credit decision would not be altered by a change of the applicant’s gender, race, religion, or medical status). One could also show that the algorithm being used comes from a particular family of algorithms, for example that a particular decision was made using one of the fair classification algorithms described in Section IV.B.2. In fact, these techniques can certify any property of the decision algorithm that can be checked by a second examination algorithm. Such properties can be proven by making the examination algorithm public and giving a zero-knowledge proof that, if the examination algorithm were run on the secret decision

---

\(^{87}\) One example is the privacy regime created by the Health Insurance Portability and Accountability Act (“HIPAA” - 110 Stat. 1936), which forbids the disclosure of certain types of covered information beyond those for which the data subject was previously given notice and which limits disclosure to covered entities subject to the same restrictions.
algorithm, it would report that the decision algorithm has the desired property.

Those who use algorithmic decision making today regularly make these types of assertions, either because they are required by law to disclose certain facts about their decision process to regulators or consumers or simply because they want to generate good will or demonstrate better behavior than a competitor. But these assertions are just words on paper, subject to challenge by a skeptical regulator or disbelief by a skeptical consumer. This skepticism is not entirely unfounded: these assertions have proved wrong in the past. Zero-knowledge proofs give a more direct connection between the fact being asserted and the technical mechanism of decision-making. This proof provides the consumer with a high assurance that the assertion proffered relates meaningfully to the facts on the ground. For this reason, proofs are also a valuable tool for compliance: designing systems to admit proofs is a helpful exercise in reasoning about how they should behave and how they come to behave that way, and is often useful as a way to discover and fix implementation errors or other compliance problems. That is, proofs are also a valuable certification to the implementer of a system that they system is working as intended.

These techniques can also be used to facilitate a kind of *ex post facto* accountability regime, under which a decision maker can demonstrate that they will be able to show how and why they used certain data after the fact in the case of audit or dispute. This can help to quell concerns that decisions are based on misuse of protected status information required by techniques for fair modeling.

C. Anti-Discrimination Law and Algorithmic Decision-Making

The goal of Part II—procedural regularity—is relatively simple from a legal standpoint. Procedural regularity is a core idea behind due process: the state cannot single out an individual for a different procedure. An argument that governance measures ensuring

88 See e.g. 12 C.F.R Part 203.4 & 203.5
89 [CITE example TK]
90 See, e.g., Arthur S. Miller, An Affirmative Thrust to Due Process of Law?, 30 Geo. Wash. L. Rev. 399, 403 (1962) (“Procedural due process (‘adherence to procedural regularity’), as we have often been told by Supreme Court justices, is the very cornerstone of individual liberties.”).
algorithmic procedural regularity are required by due process is more tenuous, but an agency that implements such measures will not risk violating a legal requirement.

In contrast, governance of algorithms to promote non-discrimination runs into the complicated field of anti-discrimination law. Here, the movement towards a colorblind interpretation of equal protection has created friction with the precedents involving disparate impact. We argue that, given the current state of anti-discrimination law, designing for non-discrimination is important because users of algorithms may be legally barred from revising processes to correct for discrimination after the fact and technical tools offer solutions to help.


Anti-discrimination law is based upon the constitutional guarantee of equal protection and on statutory measures. In recent times, interpretation of the Equal Protection Clause has been divided between those who believe in a colorblind Constitution—protecting individualized assessments and eschewing any evaluations based on group status—and those who support antisubordination—attempting to remedy inequalities between groups. The general trend has been towards colorblindness.

---

92 U.S. Const. amend. XIV, § 1 (“No State shall . . . deny to any person within its jurisdiction the equal protection of the laws.”). The Equal Protection Clause has been interpreted to apply to the federal government as well through the Due Process Clause of the Fifth Amendment. See, e.g., Kenji Yoshino, The New Equal Protection, 124 Harv. L. Rev. 747, 748 n.10 (2011).
93 See, e.g., Reva B. Siegel, From Colorblindness to Antibalkanization: An Emerging Ground of Decision in Race Equality Cases, 120 Yale L.J. 1278, 1281 (2011) (describing this binary as the common interpretation of equal protection jurisprudence).
94 See Case Comment, Civil Rights Act, Title VII—Compliance Efforts: Ricci v. DeStefano, 123 Harv. L. Rev. 282, 289 (2009) (“The Court’s conception of equal protection turns largely on its swing voter, Justice Kennedy, who appears to support a moderate version of the colorblind Constitution.”); but see Reva B. Siegel, The Supreme Court, 2012 Term—Foreword: Equality Divided, 127 Harv. L. Rev. 1, 6 (2013) (agreeing that “[s]hifts in equal protection oversight . . . are continuing to grow” but arguing that these changes are “neither colorblind nor evenhanded” because “the Court has encouraged majority claimants to make discriminatory purpose arguments about civil rights law based on inferences the
For statutory measures, we will focus on Title VII of the Civil Rights Act of 1964. Under Title VII remedies are available for disparate treatment—discriminatory intent or the formal application of different rules to people of different groups—and disparate impact—results that differ for different groups. Algorithmic decision-making blurs the definitions of disparate treatment and disparate impact and poses a number of open questions:

Is it disparate treatment when the inputs used are a proxy for membership in a protected class? Different rules are effectively applied to different groups in this case, but that difference may have no effect on the outcomes. If the people responsible for a decision know that an algorithm behaves in a way that has disparate impact, does that mean that they intend a discriminatory result? If an algorithm generates poor outcomes for a group of people, how accurate does the algorithm need to be (and how carefully does the decision-maker need to test alternative algorithms) before the decision-maker can escape disparate impact liability because the factors used are job-related? If, as noted in Section III.B.2, knowledge of class membership can be used to improve the fairness of outcomes for members of all classes, should doing so be considered disparate treatment?

These doctrines were recently considered in Ricci v. DeStefano, when the Supreme Court held that “before an Roberts Court would flatly deny if minority claimants were bringing discriminatory purpose challenges to the criminal law”).


97 See Barocas & Selbst at 24-43.

98 Id. at 24-26.

99 Id. at 30-31.

100 Id. at 36-43.

employer can engage in intentional discrimination for the asserted purpose of avoiding or remedying an unintentional disparate impact, the employer must have a strong basis in evidence to believe it will be subject to disparate-impact liability if it fails to take the race-conscious, discriminatory action.”

At issue was the City of New Haven’s test for firefighter promotions: though the tests had been constructed in an attempt to ensure non-discrimination by race, the pass rates of minorities were about half of the pass rate for whites. The New Haven Civil Service Board did not certify the results of the test (and validate the promotions) due to concerns about fairness and disparate impact liability for the City.

*Ricci* demonstrates the tensions between disparate treatment and disparate impact. Facially neutral policies can produce unequal results for protected classes, but remedying that disparate impact would require the state to treat people differently based on class membership, which *Ricci* forbids. *Ricci* also hints at the difficulties in squaring the Court’s move towards a colorblind interpretation of the Equal Protection Clause and the doctrine of disparate impact: the holding does not directly address the constitutional issue, but Justice Scalia’s concurrence does note that the “war between disparate impact and equal protection will be waged sooner or later.” Both of these doctrinal tensions are of concern to law-and policymakers.

2. *Ricci* Impels Designing for Non-Discrimination

Although *Ricci* has generated a wide-ranging conversation about equal protection, disparate treatment, and disparate impact, we wish to emphasize its implications for the governance of decision algorithms for processes where non-discrimination is a goal. The holding in *Ricci* suggests that we cannot rely on auditing for legal reasons, in addition to the reasons discussed in Section II.A. If an agency runs an algorithm that has a disparate impact, correcting those results after the fact will trigger the same kind of analysis as New Haven’s rejection of its firefighter test results. It is even possible that the Court will “subject some range of disparate

102 *Id.* at 2677.
103 *Id.* at 2665.
104 *Id.* at 2678.
105 *Id.* at 2667-71.
106 *Id.* at 2683 (Scalia, J., concurring).
impact compliance efforts to strict scrutiny," a high bar that will be difficult to satisfy in most cases.

The legal difficulties with correcting discriminatory algorithms ex post make measures to design algorithms for non-discrimination even more important. The Court in Ricci took no issue with New Haven’s process of designing the tests with an eye towards nondiscrimination: “Title VII does not prohibit an employer from considering, before administering a test or practice, how to design that test or practice in order to provide a fair opportunity for all individuals, regardless of their race.” However, “once that process has been established and employers have made clear their selection criteria, they may not then invalidate the test results, thus upsetting an employees legitimate expectation not to be judged on the basis of race.”

The uneasy fit of algorithmic decision-making into the disparate treatment/disparate impact framework does mean that someone could allege disparate treatment because the design of the algorithm includes inputs that are a proxy for class membership, and so there is a formal application of different rules to different groups of people. However, such a claim would be valid against virtually any algorithm with a significant number of inputs—it seems more likely that courts would reject the formal-rule subset of disparate treatment for algorithmic decisions than that they would hold the majority of algorithmic decision-making to be disparate treatment. In the end, incorporating non-discrimination in the initial design of algorithms is the safest path that decision-makers can take, and we should encourage the development and deployment of technical tools to aid in that design.

IV. FOSTERING COLLABORATION ACROSS COMPUTER SCIENCE, LAW, AND POLICY

In this Part, we consider the relationship between oversight as deployed in law and policy and the types of technological assurance described in Parts II and III. In technical approaches, it is traditional to have a detailed, well-defined specification of the behavior of a system in all possible cases. In lawmaking and the application of public policy, it is normal, even encouraged, for rules to be left open to interpretation, with details filled in through

107 Case Comment, Civil Rights Act, Title VII—Compliance Efforts: Ricci v. DeStefano, 123 Harv. L. Rev. at 290.
108 129 S. Ct. at 2677.
109 Id.
disputes which are resolved after-the-fact and which only give coherent detail for specific cases. We offer recommendations for dealing with this apparent mismatch, arguing for greater collaboration between experts in these different fields.

For computer scientists, we emphasize that they cannot assume that the policy process will give them a meaningful, universal, and self-consistent theory of fairness to use as a specification for algorithms; the policy process also may not accept such a theory when generated by computer scientists. For law- and policymakers, we highlight three changes that stem from algorithmic decision-making: 1) choices of algorithms embed specific policy choices, 2) algorithms can permit direct accountability to the public or other third parties, and 3) full transparency is neither sufficient nor necessary for accountability. For both groups, we note that the interplay between these areas will raise new questions and can generate new insights into what the goals of these decision-making processes should be.

A. Recommendations for Computer Scientists: Design for After-the-Fact Oversight

Computer scientists may tend to think of accountability in terms of compliance with a detailed specification set forth before the creation of an algorithm. For example, it is typical for programmers to define bugs based on the specification for a program—anything that differs from the specification is a bug; anything that follows it is a feature. However, such a mindset can conflict deeply with many sources of authority to which developers may be responsible. Public opinion and social norms are inherently not precisely specified. The corporate requirements to satisfy one’s supervisor (or one’s supervisor’s supervisor) may not be clear. Perhaps least intuitively for computer scientists, the operations of U.S. law and policy also work against clear specifications. Below, we explain in more detail why these processes often create ambiguous laws,

---

leaving details\footnote{See e.g., 47 U.S.C. § 222(c)(1) ("approval of the customer" required to use or disclose CPNI, but Congress left the FCC to define ‘approval’)}—or sometimes even major concepts\footnote{Marbury v. Madison, 5 U.S. 137 (1803) (Constitution was silent on judicial review)}—open to interpretation.

One cause of this ambiguity is the political reality of legislation. Legislators may be unable to agree on details but able to get votes to pass relatively vague language. Different legislators may support conflicting specific proposals that can be encompassed by a more general bill.\footnote{Rick Hasen, Vote Buying, 88 Calif. L. Rev. 1323, 1339 (2010)} Alternatively, legislators might not know precisely what they want but still object to a particular proposed detail; each detail causing sufficient objections would need to be stripped out of a bill before it could become law.\footnote{See e.g. 17 U.S.C. § 1201 (Congress gave the Copyright Office the power to create exemptions from the statute’s prohibition on anti-circumvention)}

Another explanation of ambiguity is uncertainty about the situations to which a law or policy will apply. Drafters may worry that they have not fully considered the space of possibilities and may want to build in enough flexibility to cover unexpected circumstances.\footnote{See e.g. 17 U.S.C. § 1201 (Congress gave the Copyright Office the power to create exemptions from the statute’s prohibition on anti-circumvention)} Incorporating this kind of give can allow a law or policy to grapple with not only current situations but also future technologies or developments that were impossible to anticipate. The U.S. Constitution is often held up as an example of this benefit: generalized provisions for governance and individual rights continue to be applicable even as the landscape of society changes dramatically.\footnote{David Straus, The Living Constitution (Oxford: 2010)}

Finally, ambiguity may stem from shared uncertainty about what tactic is best. Here, drafters may feel that they know what situations will arise but still not know how they want to deal with them. Vagueness supports experimentation to help determine what methods are most effective or desirable.\footnote{A similar logic—policy experimentation among the states—is one of the principles underlying federalism. See New State Ice Co. v. Liebmann, 285 U.S. 262 (1932) (the states “serve as a laboratory” for democracy.)}
The United States has a long history of dealing with these ambiguities through after-the-fact oversight by the courts.\textsuperscript{119} Disagreements about the application of a law or regulation to a specific set of facts can be resolved through cases, and the holes of ambiguity are filled in by the accretion of many rulings on many different, specific situations.\textsuperscript{120} Though statutes and regulations may have specific and detailed language, they are expected to be interpreted through cases—with appropriate deference given to the expertise of administrative agencies.\textsuperscript{121} Those cases form binding precedents, which in the U.S. common law system,\textsuperscript{122} are as authoritative a source of law as the statutes themselves.\textsuperscript{123} The gradual development and extension of law and regulations through cases with specific fact patterns allows for careful consideration of meaning and effects at a level of granularity that is usually impossible to reach during the drafting process.\textsuperscript{124}

The above discussion is intended to inform computer scientists that no one will remove all the ambiguities and offer them a clear, complete specification. Although law- and policymakers can offer clarifications or other changes to guide the work done by developers,\textsuperscript{125} drafters may be unable to remove certain ambiguities for political reasons as well as unwilling for flexibility objectives. As such, computer scientists must account for the lack of precision—and the corresponding need for after-the-fact oversight by courts or other reviewers—when designing decision-making algorithms.

In practice, these characteristics imply that computer scientists should focus on creating algorithms that are reviewable, not just compliant with the specifications that are generated in the drafting process.\textsuperscript{126} Concretely, this means it would be good for the

\begin{footnotesize}
\textsuperscript{119} E. Alan Farnsworth, Introduction to the Legal System of the United States (4\textsuperscript{th} ed., 2010)
\textsuperscript{120} Id.
\textsuperscript{122} See Farnsworth, supra note XX
\textsuperscript{123} Id.
\textsuperscript{124} Id.
\textsuperscript{125} See infra [IV.B].
\textsuperscript{126} Another possible conclusion is that certain algorithms should also be developed to be flexible, permitting adaptation as new cases, laws, or regulations add to the initial specifications. The need to adapt algorithms is discussed further in Section [IV.B.1]. This also reflects the insufficiency of building a system in accord with a particular specification—though oversight or
\end{footnotesize}
Diversity Visa Lottery described in Section II.C to use an algorithm that made fair random choices and it would be better for the State Department to be able to demonstrate that property to a court or a skeptical lottery participant.127

The technical approaches described in this Article128 provide several ways for algorithm designers to ensure that the basis for a decision can be verified later. With these tools, reviewers can check whether an algorithm actually was used to make a particular decision,129 whether random inputs were chosen fairly,130 and whether the algorithm comports with certain principles specified at the time of the design.131 Essentially, these technical tools allow continued after-the-fact evaluations of algorithms by allowing for and assisting the judicial system’s traditional role in ultimately determining the legality of particular decision-making.132

Implementing these changes would improve the accountability of decision-making algorithms dramatically, but we see that implementation as only a first step. We encourage research into extensions of these technical tools, as well as new techniques designed to facilitate oversight.

B. Recommendations for Law- and Policymakers

The other side of the coin is that law- and policymakers need to recognize and adapt to the changes wrought by algorithmic decision-making. Characteristics of algorithms offer both new opportunities and new challenges for the development of legal regimes governing decision-making: algorithmic decision-making can reduce the benefits of ambiguity, increase accountability to the public, permit accountability even when aspects of the decision process remain secret.

enforcement bodies evaluating the decision at a later point in time will still need to be able to certify compliance with any actual specifications.

127 Algorithms offer a new opportunity for decision-making processes to be reviewed by non-traditional overseers: decision recipients, members of the public, or concerned nongovernmental organizations. We discuss this possibility further in Section IV.B.2.

128 See supra Sections [II.B & III.B].

129 See supra notes XX-YY and accompanying text

130 See supra notes XX-YY and accompanying text

131 See supra notes XX-YY and accompanying text

132 Computer scientists model this after-the-fact input as an “oracle” that can be consulted only rarely on the acceptability of the algorithm. [CITE]
1. Reduced Benefits of Ambiguity

Although computer scientists can build algorithms to permit after-the-fact assessment and accountability, they cannot alter the fact that any algorithm design will encode specific values and involve specific rules. Furthermore, as we have explained throughout this Article, after-the-fact accountability can be limited by algorithmic design. In other words, if an algorithm is not designed to permit certification of a particular characteristic, an oversight body cannot be certain that it will be able to certify that characteristic. Both of these traits imply that algorithmic decision-making can exacerbate certain disadvantages of legal ambiguities.

In the framework set forth above, we identify three possible explanations for ambiguity: political stalemate, uncertainty about circumstances, and desire for policy experimentation. Here, for each of these cases, we will discuss how the calculation of the relative advantages of precision versus ambiguity shifts when applied to algorithmic decision-making and will offer suggestions for retaining the intended benefits of the U.S. lawmaking system.

Ambiguity stemming from political stalemate essentially passes the buck for determining details from legislators to someone later in the process. These later actors tend to be more sheltered from political pressures and thus able to make specific decisions without risking their jobs at the next election. Currently, judges and administrative agencies ideally would fill this role. Courts are expected to offer impartial decisions resistant to public pressure and administrative agencies similarly are expected to retain staff despite changes in political administrations, and those staff members also should offer subject matter expertise beyond what is expected of legislators.

However, this transfer of responsibility sometimes works in less than ideal ways. Algorithmic decision-making may exacerbate

---

133 Supra Section [IV.A].
134 See e.g. Alexander Hamilton, The Federalist Papers, No. 78, (laying out the philosophy that the judiciary’s role, as distinct from the executive and legislative powers, was to interpret the law.) However, the rise of elected judges raised questions about this traditional role of the court system. See Stephen Choi, G. Mitu Gulati, Eric A. Posner, Professionals or Politicians: The Uncertain Empirical Case for an Elected Rather Than Appointed Judiciary, U of Chicago Law & Economics, Olin Working Paper No. 357 (2007)(finding elected judges behave more like politicians than appointed independent judges.)
these problems by adding another actor to whom the responsibility can devolve: the developer who programs the decision-making software. Citron offers examples of failures in automated systems that determine benefits eligibility, notably the airport “No Fly” lists, terrorist identifications, and punishment for “dead-beat” parents.\footnote{Technological Due Process, supra note XX at 1256-57.} Lawmakers should consider this possibility and should avoid giving the responsibility for filling in details of the law to developers because 1) the algorithm will apply broadly, affecting all participants, 2) the program developer is unlikely to be held accountable by the current political process, and 3) the program developer is unlikely to have expertise about the decision being made.\footnote{Id. A distinction should be drawn here between the responsibilities given to individual developers of particular algorithms and the responsibilities given to computer scientists in general. Great gains can be made by improved dialogue between computer scientists and law- and policymakers about how to design algorithms to reach social goals. See infra Part IV.C} One potential method for restricting the discretion of developers—without requiring specifications in the legislation itself—would be to legislate guidance for software development by administrative agencies. Difficulties in translating between code choices and policy effects still would exist, but could be eased using the technical methods we have described.\footnote{See Parts II.B & III.B.} For example, administrative agencies could work together with developers to identify the properties they want the algorithm to possess, and the algorithm then could be designed to permit proof that it satisfies those properties.

Ambiguity generated by uncertainty about circumstances or by a desire for policy experimentation presents a more complex concern. Here, the problem raised by algorithmic decision-making is that the algorithm locks in a particular choice of code for the duration of its use, and especially in government contexts, provisions may not be made to update the code. Worries about changing or unexpected circumstances could be assuaged by adding sunset provisions to algorithms,\footnote{The effectiveness of sunset provisions in leading to actual reconsideration and change is debatable. The inertia of the pre-existing choices can be hard to overcome. See e.g. Mark A. Lemley & David McGowan, Legal Implications of Network Economic Effects, 86 Calif. L. Rev. 479 (1998)} requiring periodic review and reconsideration of the software. Algorithms additionally could be designed with eventual revisions and updates in mind. As for preserving the benefits of policy experimentation,
the traditional solution might be having multiple algorithms. A more sophisticated version of this solution is the incorporation of machine learning into decision-making systems. Again, machine learning can have its own fairness pitfalls, and care should be taken to consider fair machine learning methods and to build in precautions like persistent testing of the hypotheses built into the machine learning model.

More generally, the benefits of ambiguity decrease in the case of algorithmic decision-making. Here, an uninformed programming actor may determine the details and then applied them broadly. In addition, the choice of algorithm cements the particular policy choices encoded in that algorithm for as long as it is used. Drafters should consider whether they instead should increase the specificity offered by law and policy governing these algorithms.

To a certain extent, this question mirrors the rules versus standards debate about the relative merits of laws that specify actions and their repercussions (for example, a speed limit) and those that espouse a principle open to interpretation (for example, “drive at a speed reasonable for the conditions”). Rules give clarity and forewarning, while standards offer greater flexibility for interpretation.

Here, the question is whether drafters should include additional and clearer specifications for developers. In practice, drafters may wish to incorporate a set of narrow rules within a broad, overarching standard. For example, drafters could include 1) specifications of each of the properties that they want to ensure that an algorithm possesses and requirements that the developer design the algorithm in a way that renders those properties provable upon review alongside 2) a general statement of purpose for the algorithm. Doing so would give the developer some flexibility in writing the code while also ensuring that particular properties can be checked later.

---

140 See supra note XX
141 See supra note XX
142 In other words, even after a machine learning algorithm determines that particular rule should be used to produce particular results, it always should continue to test inputs that do not follow that rule.
143 See e.g. Louis Kaplow, Rules versus Standards: An economic analysis, 42 Duke L.J. 557 (1992)
144 See Kathleen Sullivan, Forward: The Justices of Rules and Standards, 106 Harv. L. Rev. 22 (1992)
2. Accountability to the Public

Oversight is traditionally performed by courts, enforcement agencies or other designated entities such as government prosecutors. Typically, the public and third parties have an indirect oversight role through the ability to provide political feedback and the ability to bring lawsuits if their specific circumstances allow. The use of algorithms can alter how effectively the legal system and the public can oversee the decision-making process.

In one sense, decision-making algorithms can enhance accountability to the public and interested third parties by permitting greater involvement in oversight. The technical tools we describe allow for a more direct form of oversight by these parties. Unlike traditional legal oversight mechanisms that generally require discovery or the gathering of internal evidence, the technical tools may enable verifications by the public and third-parties that are not completely independent from the organizations using the algorithms. For example, technologically proficient members of the public or third parties could perform the verifications that a particular algorithm was used or that it has particular properties. In addition, a system could be built for participants to check these properties for their own outcomes so that non-technical users could perform these verifications, while the system itself would be overseen by others—potentially both inside and outside of government—who do have the needed technological expertise. As another example, third parties could be involved in generating fair randomness.

In contrast to the possibility for enhanced public accountability, the use of these algorithms and the reliance on technical tools for oversight can also reduce accountability to the public by hampering the traditional court-based scrutiny of decision-making. The U.S. court system is designed to protect against wrongful government actions through the power of judicial review. Judicial review gives judges the power and responsibility to determine if government actions comply with legal obligations. Similarly, for private actions, the legal system

---

145 See Farnsworth, supra note XX.
146 The public can vote political leaders out of office and aggrieved parties can bring law suits to seek vindication.
147 See supra notes XX.
148 See Marbury v. Madison, 5 U.S. 137 (1803)
vests judges and regulatory agencies with the authority to determine whether those actions are consistent with legal standards.

However, the use of technical tools shifts the determination of regularity from the courts and enforcement agencies to other parties, specifically external experts or the organizations using the algorithms themselves. This arises because the courts and enforcement agencies are no longer making the determination whether the rules have been properly applied. The determination shifts to the experts evaluating the algorithmic decision-making process. One way to address this unintended shift in responsibility might be to appoint technical experts as special masters. Courts typically appoint special masters to perform functions on behalf of the court that require special skill or knowledge.149

Another issue that challenges public accountability is the validation of the technical tools. For courts, technical tools cannot be accepted until their integrity and reliability are proven. Courts, though, have long confronted the problem of the admissibility of scientific evidence. For example, the courts took years during the 1980s and 90s to establish and accept the scientific validity of DNA and the methods used to isolate DNA.150 The Federal Rules of Civil Procedure now provide for the acceptability of new scientific methods in adversarial proceedings.151 In 1993, the Supreme Court set out standards to meet the Federal Rules requirements that include testing, peer review and publication.152 This addresses the validation of technical tools used to examine algorithmic decision-making, but still leaves open the assurance of the technical tools’ reliability. Ordinarily, the U.S. legal system relies on the adversarial process to assure the accuracy of findings. This attribute may be preserved by allowing multiple experts to test algorithmic processes.

149 See e.g. U.S. v. Microsoft, 147 F. 3rd 935 (1998)(the court appointed Larry Lessig to serve as a special master for technical issues associated with the anti-trust case brought against Microsoft)
151 See Fed. R. Civ. Pro, Rule 702 (“If scientific, technical, or other specialized knowledge will assist the trier of fact to understand the evidence or to determine a fact in issue, a witness qualified as an expert by knowledge, skill, experience, training, or education, may testify thereto in the form of an opinion or otherwise.”)
3. Secrets and Accountability

Implementing algorithmic decision-making in a socially and politically acceptable way requires advances in our ability to communicate and understand fine-grained partial information about how decisions are reached. Full transparency (disclosing everything) is technically trivial but politically and practically infeasible nor always useful as described in Section II.A. However, disclosing nothing about the basis for a decision is socially unacceptable and generally poses a technical challenge. Law- and policymakers should remember that it is possible to make an algorithm accountable without the evaluator having full access to the algorithm.153

U.S. law and policy often focuses upon transparency and even equates oversight with transparency for the overseer.154 As such, accountability without full transparency may seem counterintuitive. However, oversight based on partial information occurs regularly within the legal system. Courts prevent consideration of many types of information for various policy reasons: disclosures of classified information may be prevented or limited to preserve national security;155 juvenile records may be sealed because of a decision that youthful mistakes should not follow one forever;156 and other evidence is deemed inadmissible for a multitude of reasons, including being unscientific,157 hearsay,158 inflammatory,159 or illegally obtained.160 Thus, precedent exists for basing oversight on partial information.

Strong policy justifications exist for holding back information in the case of algorithmic decision-making. Revealing algorithms and their input data can expose trade secrets, violate privacy, hamper law enforcement, or lead to gaming of the decision-making process.

153 See supra note XX [n. 52]
154 See e.g. 5 U.S.C. § 552 (Freedom of Information Act requires access to government held information); 15 U.S.C. § 6803 (GLBA requires financial service providers to provide annual privacy notices as a transparency measure.)
156 See e.g. N.Y. Crim. Pro. § 720.15 (providing for filings under seal in juvenile proceedings.)
157 Fed. R. Civ. Pro, Rule 702 (establishing the court’s discretion to admit scientific evidence)
158 Fed. R. Civ. Pro, Rule 802 (excludes evidence from hearsay)
159 Fed. R. Civ. Pro, Rule 403 (excludes evidence for prejudice)
160 See e.g. 18 U.S.C. § 2515 (exclusionary rule for evidence obtained through wire tap or interception)
process. The advantage of algorithms is that concealment of code and data does not imply an inability to analyze that code and data. The technical tools we describe give law- and policymakers the ability to keep algorithms and their inputs secret while still rendering them accountable. They can apply these tools by implementing them in government-run algorithms, such as the DVL, and to incentivize non-governmental actors to use them, perhaps by mandating that use or by requiring transparency—at least to courts—of algorithms and inputs if they do not employ such technical tools.

161 See Part II.A