The version of record is available at: http://dx.doi.org/10.1561/3300000002

Accountability in Computing: Concepts and Mechanisms

Joan Feigenbaum[1], Aaron D. Jaggard[2] and Rebecca N. Wright[3]

[1] *Yale University; joan.feigenbaum@yale.edu*
[2] *U.S. Naval Research Laboratory; aaron.jaggard@nrl.navy.mil*
[3] *Barnard College; rwright@barnard.edu*

ABSTRACT

Accountability is a widely studied but amorphous concept, used to mean different things across different disciplines and domains of application. Here, we survey work on accountability in computer science and other disciplines. We motivate our survey with a study of the myriad ways in which the term "accountability" has been used across disciplines and the concepts that play key roles in defining it. This leads us to identify a temporal spectrum onto which we may place different notions of accountability to facilitate their comparison. We then survey accountability mechanisms for different application domains in computer science and place them on our spectrum. Building on this broader survey, we review frameworks and languages for studying accountability in computer science. Finally, we offer conclusions, open questions, and future directions.

Joan Feigenbaum, Aaron D. Jaggard and Rebecca N. Wright (2020), "Accountability in Computing: Concepts and Mechanisms", Foundations and Trends® in Privacy and Security: Vol. 2, No. 4, pp 247–399. DOI: 10.1561/3300000002.

The version of record is available at: http://dx.doi.org/10.1561/3300000002

1

Introduction: The Problem of "Accountability"

1.1 Motivation

The best known and most widely deployed security technologies—*e.g.*, passwords, authentication protocols, firewalls, and access-control mechanisms—are *preventive* in nature. The idea is to stop unauthorized parties before they can download confidential data that they are not supposed to have access to, login to a proprietary network of an organization that they do not belong to, or take any other action that violates system policy. However, dramatically increased scale and complexity of Internet commerce, social networking, remote work, distance learning, and myriad other forms of social, economic, and intellectual engagement online with both strangers and friends has rendered preventive mechanisms inadequate. The result is growing interest in *accountability* mechanisms to complement preventive measures.

Despite widespread agreement that "accountability" is needed if online life is to flourish, the term has no universally accepted definition. However, the concept has been studied extensively, both in computer science and in other disciplines. Our purpose in this monograph is to survey and contextualize these investigations, to identify key ideas, and to suggest interest directions for future research.

An essential premise for the study of accountability in computer science is that it is a natural approach to the design and implementation of security and privacy in computer systems. After all, it is a combination of preventive security measures and after-the-fact accountability with which rules have always been enforced in the offline world. We offer three examples of real-world scenarios in which after-the-fact accountability mechanisms complement preventive security in essential ways.

Digital copyright: US copyright law is a clear example of a set of policies that cannot be enforced in a purely preventive manner if they are to achieve their goals. One bedrock copyright principle is that the creator of a copyright work has certain exclusive rights, including the right to control copying and distribution of his work and the right to authorize or refuse to authorize the creation of derivative works (such as sequels or movie versions of books). However, the law also specifies exceptions to these exclusive rights in the form of fair-use provisions. Under the fair-use doctrine, a researcher, for example, may make a small number of copies of a scientific journal paper for use by the members of his lab without obtaining the author's permission, but he may not (without the author's permission) share the article with everyone at his university or some other wide audience. A properly attributed excerpt from a newspaper story may, without the author's permission, be copied and distributed widely without infringing copyright or committing plagiarism. The notion of fair use promotes socially desirable activities, such as education and criticism, and is regarded by many as an essential pillar of cultural production.

In the analog world of books, magazines, newspapers, and academic journals, there is no attempt at preventive enforcement of copyright law. It is technologically feasible to violate the law by making and distributing many unauthorized copies of a book, but anyone who does so runs the risk of being caught and sued for copyright infringement (*i.e.*, being "held accountable" for an illegal actions), and in any case one incurs the nonnegligible cost of copying and distribution. The fact that copyright enforcement is based on detection rather than prevention supports fair use. To determine whether, and if so how, one wants to use a document and whether such a use requires the author's permission,

one must be able to read (and, in particular, to have access to) the document; preventive copyright enforcement might restrict access to those who could justify that access *a priori* and those who are willing and able to pay for access.

Digital creation and distribution of books, songs, movies, *etc.* has motivated attempts at preventive copyright enforcement. Digital-Rights-Management (DRM) systems are justified in part by the negligible cost of copying and distributing digital works. Unfortunately, some DRM systems impose severe limits (or even prohibition) on uncompensated use of the works they manage. Crafting these limits in a manner that is consistent with the goals of copyright law is certainly hard and may be impossible; if the limits are very strict, they threaten fair use, but, if they are too permissive, the works might be too easily copied and distributed and the creators' rights vitiated. We believe that access and accountability together form a better approach to digital copyright than draconian forms of preventive DRM. Allowing users to access digital copyright works, just as they access analog works when browsing in physical stores and libraries, is consistent with enforcement of creators' rights provided that they are held accountable for subsequent use of those works in accordance with copyright law.

Break-glass scenarios: In some emergency situations, there is a clear need to augment or complement traditional, preventive access controls and usage policies. They are often called *break-glass scenarios*—a reference to the fact that one often must break a glass cover in order to pull a fire alarm.

For example, a physician in one medical practice may not, as a routine matter, have access to the patient records of another medical practice. If that physician encounters a patient of the other practice who needs emergency treatment, she could present her medical ID and a description of the emergency to the other practice and be granted temporary access to the patient's records. The information she provides to the emergency-care team should be logged, and all emergency-access logs should be audited periodically. If a doctor is discovered to have used her medical ID in this manner when not in a true emergency, the legal and professional penalties would be significant. The combination

of secure logs and substantial penalities should deter abuse of the emergency-access system and thus support patients' privacy—in other words, physicians would be "held accountable" for proper emergency use of patients' records.

Credit-card authorization: Retail use of credit cards is an excellent example of how accountability and authorization can prevail without strong authentication or even a conventional notion of identification. Thus, it supports our contention that "accountability" is not simply a matter of identifying all participants in a system, keeping track of all of their actions, and punishing actors who break the rules.

If Jane, a customer, attempts to pay for something in a store with her neighbor Mary's credit card, and Mary's card is far enough below its spending limit to accommodate the purchase (and has not been reported stolen), then Jane is unlikely to encounter any objections by the merchant. So has the credit-card authorization system functioned properly? Detailed examination of the process is instructive.

First, note that the merchant is the resource controller in this example. It is he who seeks assurance that, after a customer leaves his store with an item, he will be paid for that item. The system that he uses to obtain this assurance has both a technological component and a legal component: He can swipe the offered card, enter the price of the item and, after a few seconds, receive official approval or rejection of the purchase; if the purchase is approved, then the card issuer is legally obligated to pay him, regardless of any dispute that may subsequently arise between the issuer and the cardholder. (Presumably, if such a dispute had already arisen, the purchase would not be approved.)

Thus, whether it is the cardholder (Mary in our example) or someone else (*e.g.*, her neighbor Jane) who presents the card does not matter to the merchant—the resource that he controls is his store inventory, and the payment stream that he cares about is the one from the card issuer to him. Control over the card as a resource and concern about payment by the cardholder is the concern of the card issuer, not the merchant.

Note that the correctness conditions for which parties might be held accountable are not necessarily directly aligned.

Merchant: As noted above, the merchant desires assurance that, if an item leaves the store, he will receive its price.

Customer: The customer does not want to become obligated to pay any more than the price of the items she takes from the store.

Card issuer: The credit-card issuer desires assurance that, if he becomes obligated to the merchant for some amount, then some customer becomes obligated to the card issuer for at least that amount.

This example exposes the existence of legitimate confusion and some of the questions that are essential to ask when we look at an accountability problem. Who is accountable, and for what? To whom (if anyone) are they accountable? We elaborate on this legitimate confusion in the next section.

1.2 Why "accountability" is hard to pin down

Why is the concept of accountability is so elusive? Why has the term been defined in so many different ways (some of them mutually contradictory), particularly by computer scientists, and why are some uses of the term considered counter-intuitive or misleading? We ask these questions in order to help clarify the scope of our survey, not because we expect to resolve all of the terminological confusion surrounding accountability. Following Koppell (2005), we "do not suggest a new, all-encompassing definition of the word. There are enough already!"

We believe (and provide evidence in this section) that there is no agreement on whether or not accountable systems require certain basic system properties, *e.g.*, persistent identities of system participants, public identification of alleged policy violators and formal adjudication of allegations, or quantifiable punishment of those proven to be violators.

As explained in Sec. 1.1, computer scientists have traditionally approached information security through prevention: Before taking any security-sensitive action, an entity is expected prove that it is authorized to do so. In online life, which is characterized by enormous scale and complexity, the purely preventive approach to security has proven to be insufficient. Several researchers, including Weitzner, Abelson, Berners-Lee, Feigenbaum, Hendler, and Sussman (Weitzner *et al.*, 2008), Lampson (2009), and Datta (2014), have suggested that the preventive approach be augmented by an accountability approach. Our goal in

writing this survey is to focus attention on the broadest possible class of information systems that take an accountability approach—roughly speaking, on systems in which policy violations are punished. Following Weitzner *et al.*, we call these *accountable systems*. Traditional preventive measures are not precluded in accountable systems. However, when such a system cannot simply prevent all policy violations by using passwords, authentication protocols, and other classic security mechanisms, it *ensures that users who violate system policy incur negative consequences*. Both conceptually and pragmatically, this is a natural approach to the design and implementation of policy-governed information systems; after all, it is a combination of before-the-fact prevention and after-the-fact punishment with which laws and policies have always been enforced in the offline world.

1.2.1 Responsibility, adjudication, and sanctions

Lampson (2005a) put forth a simple but apparently quite general formulation of the term in the context of information systems: "Accountability is the ability to hold an entity, such as a person or organization, responsible for its actions." This formulation is similar to the one in earlier work in political science by Grant and Keohane (2005), who say that accountability "implies that some actors have the right to hold other actors to a set of standards, to judge whether they have fulfilled their responsibilities in light of these standards, and to impose sanctions if they determine that these responsibilities have not been met." However, the differences between these two formulations are typical of the difficulties one encounters in this area.

For example, note that Grant and Keohane speak of the *right* of some actors to behave in certain ways in order to hold others responsible, while Lampson speaks of the *ability* to hold others responsible. This difference may reflect the disciplines within which the research was conducted. Rights are a central focus in political science, and it is perfectly natural to assign particular rights to entities in a political system even if those entities do not have the ability to exercise said rights. In computer science, the focus is on the capabilities and limitations of various entities in a system and on the interactions among entities, and

it is not clear why one would be interested in an entity's having the right to do something if it did not have the technical ability to do it.

Note further that Grant and Keohane assume a system in which there are at least two actors, say A and B, one of whom (say B) is accountable to the other; moreover, A has the right both to judge whether B has fulfilled his responsibilities and to impose sanctions on him if he has not. Lampson does not assume that judgment and sanctions are performed by the same entity. Indeed, he does not say anything about judgment or sanctions: Accountability in his formulation is a system property; whether people and organizations in the system must be explicitly judged and, if so, whether the entity that judges them is the same as the one that holds them responsible is not specified.

1.2.2 Automatic *vs.* mediated punishment

In earlier work, we have formalized accountability so as to include accountable systems in which there are no explicit adjudication procedures (Feigenbaum *et al.*, 2011). What is required in an accountable system is that entities that violate system policies are punished, by which we mean that a violator's *utility* is lowered as a result of the violation. Our formal framework treats in a unified manner systems in which particiants are punished *automatically* and those in which punishment is *mediated* by a judge. Participants are punished automatically when the very act of violating a system policy causes their utility to decline.

Automatic punishment of this form need not expose the identity of a violator or the nature of his violation to the rest of the system participants; in fact, the violator himself need not be aware that he has violated a system policy. The key feature of this unified framework is that, because a violator's utility is decreased as a result of his violating a system policy, participants' actions are tied to consequences. Thus, we have advocated shifting focus from the *procedures* used by accountable systems (where it is in, *e.g.*, the Grant and Keohane formulation) to the *meaning* of accountability and, specifically, what accountability mechanisms must provide if they are to be a useful complement to preventive mechanisms.

The idea of a mechanism that imposes consequences for policy violations automatically, without exposing the violation to system participants (or even to the violator), may seem odd in a computer-security context, but it is actually standard in several fields, including economics. In a "truthful" (or "strategyproof") sealed-bid auction, a bidder's utility is (provably) maximized by his honestly bidding the maximum price that he is willing to pay for the item. A bidder who violates the "bid your true value for the item" policy may wind up losing utility because he wins the auction but pays more than he actually thinks the item is worth or because he loses the auction to someone else who did not value the item as highly as he did. The auction mechanism imposes consequences upon policy violators automatically in the process of choosing a winner and setting a price.

The main objection to our classifying this standard economic notion of *incentive compatibility* and other automatic-punishment mechanisms as forms of accountability is that they do not necessitate "calling the violator to account" or "making him account for himself" that popular usage of the term connotes and that some theorists of accountability, *e.g.*, Mulgan (2000), have identified as a core component. In that view of accountability, there must be *social exchange* between an accountable entity and the entity calling for the account, and there is social value in a *public accounting* that makes clear to all entities in the system the nature of the violation and the consequences that attach to it. The objection is not that automatic punishment without public accounting has no value but rather that it is not properly regarded as an "accountability" mechanism; some have suggested that *deterrence* is a better term for the system property that such mechanisms provide.

1.2.3 Identity, anonymity, and pseudonymity

The Grant–Keohane conception of accountability presented in Sec. 1.2.1 seems to assume that participants in accountable systems have persistent identities and, in particular, that they are identifiable by those who have the right to hold them accountable. But online interaction is sometimes anonymous or at least pseudonymous, and this characteristic of online life is highly valued by many. Indeed, the intuition that

support for "accountability" in online life necessarily implies opposition to anonymity and pseudonymity has caused many cyber-rights advocates to be suspicious of the quest for accountability. Because accountability and anonymous and pseudonymous interactions are all worthwhile goals, we ask whether they must be in tension. As usual, that depends on what one means by "accountability."

In computer science, different researchers have taken a wide range of approaches to participants' identities in accountable systems. For example, several influential experimental works (*e.g.*, Andersen *et al.*, 2008; MIT Decentralized Information Group, 2009; MIT Decentralized Information Group, 2010; MIT Decentralized Information Group, 2011) require that system participants have persistent identities that are known to those who hold them accountable. Anonymous participation in such an accountable system is not possible. Other influential research (*e.g.*, Camenisch and Lysyanskaya, 2001; Camenisch *et al.*, 2007; Chaum, 1982; Corrigan-Gibbs and Ford, 2010) exemplifies a completely different (and incompatible) view of the role of identity in accountable systems; in these works, a participant is held accountable precisely in the sense that, under normal circumstances, he is anonymous, but his identity can be exposed if he violates the prescribed security policy or protocol.

We believe that neither of these approaches is sufficiently general for the plethora of online interactions in which a robust notion of accountability is desirable. Our notion of accountable systems in which punishment is automatic is fully consistent with anonymous participation. More generally, we have explored accountable systems in which participants may be bound to their system identities with varying degrees of strength as a condition of participation (Feigenbaum *et al.*, 2014).

1.2.4 Concepts *vs.* terminology

Adjudication and identification are just two of many concepts whose relationships to "accountability" are handled differently by different researchers in multiple disciplines. In the following chapters, we explore these relationships and their implications for the power and limitations of accountable systems. This exploration will provide an opportunity to

clear up, to some extent, the terminological confusion that has bedeviled the study of accountability.

However, as explained above, our primary focus is not on perfecting a taxonomy of security properties. Rather, our goal is a comprehensive exploration of techniques that can usefully complement traditional preventive security mechanisms in that they can enable punishment of policy violations in a variety of realistic scenarios. Ultimately, some of these techniques may be termed "detection," "deterrence," "incentive compatibility," *etc.*, but they are within scope of this investigation if they are not purely preventive and are potentially useful in the search for systems and applications that are more secure, more usable, and more compliant with agreed-upon policies.

We conclude this section with an eloquent cautionary note from Charles Raab (2012, p. 24): "[T]he short message is that 'accountability' is not a term to be trifled with, or used casually and rhetorically, or as a fashion accessory."

1.3 Remarks on vocabulary

Many volumes have been written about *secure systems*—roughly speaking, systems in which policy violations cannot occur, because preventive security measures stop the participants from committing them. In this volume, we consider *accountable systems*—roughly speaking, systems in which policy violations, when they occur, are punished. We use the word "system" to mean an application or network protocol (*e.g.*, an auction service or a multicast protocol) that has a "goal" (determining winners and prices or delivering content to all the subscribers, respectively). The system "policy" specifies how the participants are supposed to behave, and the accountability "mechanism" ensures—or at least facilitates—their punishment if they violate the policy. Note that accountable systems may also use preventive measures (*e.g.*, passwords or authorization protocols) to stop certain policy violations from occurring, but they operate under the assumption that some policy violations might occur and seek to impose consequences when they do.

Participants (also referred to as "principals" or "actors") in an accountable system are computational agents that may represent human

beings. Some are good actors, meaning that they always comply with the system policy, and some are bad actors (attackers, intruders, or other miscreants), meaning that they at least sometimes violate system policy. The purpose of preventive measures is to stop bad actors from accessing the system; when that is not possible, the purpose of accountability mechanisms is to impose, or enable the imposition of, consequences on bad actors. One important aspect in which accountable systems differ is in whether the participants have persistent "identities" that are known to other participants or, rather, may participate anonymously. Note that, because participants are computational agents, one flesh-and-blood human may be represented in the system by more than one agent and thus may have more than one identity.

In some accountable systems, all actors are equivalent, but in others they play different roles. For example, in keeping with the common-parlance meaning of the word "accountability," it may fall to one actor, in his role as a judge, to call to account another actor, who has been accused of a policy violation. In his attempt to defend himself, the accused may provide "evidence" of the actions that he has taken or "credentials" that prove that he is authorized by the system policy to have taken those actions.

Note that we refer to a design as a "system" if actors participate in it for reasons other than "accountability," *e.g.*, to share updates with acquaintances or to compute a function, and we refer to it as a "mechanism" if participation is for the express purpose of providing accountability-related properties. In a small number of works that we consider, it may not be immediately clear which term applies to a particular design, because actions taken by the participants may serve *both* to accomplish goals such as sharing updates or computing functions *and* to provide accountability. We will identify the cases in which accountability is intertwined with system goals in this fashion.

1.4 "Accountability" implicates many areas of computer science

The study of accountability touches upon many topics in computer-science research, all of which have their own specialized vocabularies. Examples include:

Distributed computation: Accountable systems are, in general, distributed systems in the sense that there are multiple participants and that they use multiple computers. Similarly, accountability mechanisms are typically implemented as distributed algorithms that run on multiple computers.

It is important to distinguish between distributed computations that are *centralized* and those that are *decentralized*. In the centralized case, there is unified administrative control over the entire computation. Different processes may run on different machines, but they do not make independent decisions or respond to competing incentives; in other words, all of the participants are part of the same organization or *administrative domain*. In the decentralized case, different participants not only use different machines but can be organizationally or economically separate as well; they make strategic decisions independently of each other and may have competing interests.

Because most interesting distributed systems and networks are *asynchronous*, they present technical challenges for accountability mechanisms that rely on *tamper-evident* logging to preserve evidence. Definitions of accountability in asynchronous distributed systems draw on work in *fault detection*, focusing on guarantees that violations are *eventually* detected and that valid evidence cannot be created against policy-compliant participants.

Logic and language: Proofs and evidence are intrinsic to the goals of many accountability mechanisms. Participants may be called upon to prove that they fulfilled all of their responsibilities (as defined by the system policy), to prove that someone else violated the policy, to prove that evidence was acquired at a certain time and has not been tampered with, *etc.* Even mechanisms that do not demand fully rigorous mathematical proofs often rely on interactions among participants that benefit from formal reasoning. Therefore, researchers have proposed a number of proof logics and programming languages for specifying, implementing, and reasoning about accountability mechanisms. Many of these contributions draw on previous work on *modal logic* (particularly *temporal logic*), *belief logic*, and *causality*. Important distinctions among these logics and languages include whether or not proofs are fully

automated or require human intervention and whether they enable the identification of *actual causes* and the participants responsible for them or the weaker notion of a set of *all possible causes*.

Game theory: Key elements of several approaches to accountability include blame and punishment. How to "punish" participants for a policy violation obviously depends on the nature of the accountable system in which they are participating. In systems that feature an intrinsic notion of *utility*, *e.g.*, those in which participants accumulate points or exchange money and maintain "bank" accounts, a natural way to punish a violator is to reduce his utility. One challenge that arises in this approach is the need to ensure that the punishing action that reduces a participant's utility is *causally* linked to the decision that he committed a violation; a mechanism cannot be said to have held a violator accountable if he loses utility because of some unrelated "bad luck" that he experiences after the violation. Another challenge is the question of whether punishment should be *targeted* or *collective*; a system in which all participants' utilities are reduced significantly whenever a violation occurs may deter violations, but collective punishment does not satisfy most people's intuitive understanding of an "accountability" mechanism.

Game-theoretic models have also been used in the study of accountability mechanisms that rely on *auditing*. For example, in *audit games*, the standard security-game framework (in which a defender chooses how to invest in defense, and an attacker chooses which systems to target) is enhanced with a notion of costly punishment. Audit games can be used to develop an efficient algorithm that determines an approximately optimal strategy for auditing.

We discuss these and other connections between accountability and incentives in Sec. 4.3.2

Cryptography: Like logic and languages, cryptographic techniques are useful in accountability mechanisms that need to construct evidence or proofs. *Signatures*, *timestamping*, *encryption*, *hashing*, and *authentication codes* are examples of cryptographic operations that allow participants who acquire data that they need to use as evidence (of a

policy violation or of policy compliance) to ensure that it is preserved in a confidential, tamper-evident fashion and to tie it securely to appropriate meta-data, *e.g.*, the time it was discovered or the identity of the discoverer.

Although explicit punishment plays an important role in some accountable systems, there are others in which *identification* of a policy violator is, by itself, considered an accountability mechanism. Often, there is a tacit assumption that, once identified, a violator will be expelled from the system; expulsion may be regarded as a qualitative form of punishment, in contrast to the quantitative punishments used in utility-based accountable systems. *Accountable anonymity* plays a crucial role in several applications, including digital-cash and group-commuication protocols; it guarantees that participants who follow the rules can remain anonymous but that those who deviate from the roles will have their identities revealed (at least with nontrivial probability).

Privacy-preserving, aggregate reporting has been proposed as an appropriate accountability technique in law-enforcement, surveillance, and other scenarios in which a government agency must act in secrecy but is required to have proper authorization and to follow rules. For example, a law-enforcement agency may be required to make public the approximate number of wiretaps that it conducts each year but not to reveal the identities or locations of the subjects of those wiretaps or the ongoing investigations for which it conducted them. The cryptographic technique of *secure, multiparty computation (SMPC)* has a natural role to play in this type of reporting. Individual, authenticated officers can submit required information about their wiretapping activities in an encrypted (or other privacy-preserving) form, and an SMPC protocol can check that all activities are in compliance with the applicable laws and procedures and compute the total number of wiretaps without revealing any details about subjects, locations, or ongoing investigations.

Formal methods: As we explain in Sec. 1.2 and Chap. 2, there are many plausible definitions of "accountability," some of which are quite subtle. In some frameworks, accountability must be interpreted in the context of a specific accountable system. For example, if one's general notion of accountability focuses on an actor's acquiring evidence

that can convince a judge of another actor's participation, a natural interpretation in the context of a certified-email system is that the recipient gets the email and the sender gets evidence that the recipient received the email. Formal methods such as *proof logics* and *theorem provers* are useful in precisely specifying properties that capture such context-specific interpretations (see Sec. 4.2.1). They can also be used for *automatically* or *semi-automatically* proving that the system has that property.

Accountability mechanisms that use logging and auditing are good candidates for formal methods. It is quite natural to try to achieve accountability by logging every action taken or message sent by a system participant, preserving the information in a tamper-evident manner, and examining it for proof of a violation and identification of the violator after an accusation is made or an alarm is raised. Unfortunately, capturing literally all of the information may be infeasible or may result in logs that are too voluminous to be audited in the time available. Automatically or semi-automatically producible proofs that systems that log more selectively preserve enough evidence to prove the desired accountability properties are highly desirable.

Of course, the notion of accountability has also been studied extensively in disciplines other than computer science. We address some of the similarities and differences between computer scientists' ideas on the subject and others' in Sec. 3.6.

1.5 Overview of contributions

Chapter 2 surveys *accountability-related concepts*. We present categorizations in terms of time, information, and action in Sec. 2.1; in particular, we identify a temporal spectrum of accountability goals that will prove useful in understanding work in the area: prevention, violation, detection, evidence, judgment or blame, and punishment. We subsequently use these categorizations to analyze different accountability mechanisms. In Sec. 2.2, we survey different definitions of "accountability" that are both explicit and implicit in the literature and categorize them according to their focus. The remainder of Chap. 2 discusses other accountability-related concepts.

Chapter 3 surveys *accountability mechanisms*, both implemented and proposed. Of course, the type of "accountability" provided by different mechanisms varies. Describing all of the technical details of the proposed mechanisms would be lengthy and likely not particularly useful. Instead, we identify the major distinct approaches to accountability and selected proposals that exemplify these approaches and locate them on the temporal spectrum put forth in Chap. 2. We categorize mechanisms according to how they achieve the accountability properties they provide, largely paralleling the categorization of definitions in Sec. 2.2. We summarize in Sec. 3.5 the properties of these accountability mechanisms through the lens of the time/information/action framework of Sec. 2.1. Sec. 3.6 focuses on mechanisms that have been proposed and studied in disciplines other than computer science.

Chapter 4 surveys *languages and frameworks for the study of accountability.* They are more abstract than the technical accountability mechanisms considered in Chap. 3. Languages and frameworks provide ways to describe or reason about accountable systems and accountability-related properties. We also present technical results on accountability and identity in Chap. 4. Here, the focus is on research in which accountability itself is the subject, as opposed to work that seeks to achieve a particular type of accountability in a particular context.

Finally, based on consideration of the material in this survey, Chap. 5 summarizes key ideas in the accountability literature, identifies key papers, and suggests directions for future research.

2

Perspectives, Definitions, and Concepts Across Disciplines

In this chapter, we identify broad themes that arise from considering accountability in computer science and other disciplines. We then synthesize some new perspectives on accountability that reflect this interdisciplinary point of view.

First, we argue that it is helpful to analyze accountability mechanisms in terms of the *time* at which they act, the *information* that they use, and the nature of the *action(s)* that they perform. The temporal spectrum that we identify separates different stages that are often conflated. It highlights *punishment* as the end goal and illuminates the fact that punishment is often implicitly assumed in mechanisms that operate at other times on this spectrum.

Second, we survey definitions of accountability. Some of them are formally stated in the literature, while others are implicit. We categorize definitions according to the various foci that emerge from our review: detection, evidence, identification or association of actions with the principals who perform them, answerability, blame, and punishment or deterrence.

Third, we discuss accountability-related concepts and terminology. Much of this discussion focuses on the relationships among principals,

their identities within a system, and their actions. Other work that we review considers the relationships of accountability to the concepts of transparency and causality.

2.1 Time, information, and action

As we survey approaches, we evaluate how they address three broad aspects of accountability: time, information, and action. We typically consider accountability with respect to a policy violation. The "time" aspect considers when the accountability mechanism is invoked relative to the time of the violation. The "information" aspect concerns what is known and by whom, and the "action" aspect concerns what is done and by whom. This analysis refines the one that we did with Xiao (Feigenbaum *et al.*, 2012).

2.1.1 The time aspect and mechanism goals

Different accountability mechanisms are focused on different times relative to a policy violation; often this reflects the fact that different mechanisms have different goals. For example, the formal framework of Küsters, Truderung, and Vogt (Küsters *et al.*, 2010) explicitly models and focuses on judgments or verdicts, *i.e.*, declarations that a system participant is guilty of committing a violation. By contrast, the formal framework that we presented in (Feigenbaum *et al.*, 2011) focuses on punishment, which, in other frameworks, typically follows a declaration of guilt. Within our punishment-focused framework, however, there need not be an explicit judgment that identifies an individual principal as guilty; so the punishment focus is indeed distinct from the judgment focus.

Motivated by this type of distinction, we have identified (Feigenbaum *et al.*, 2012; Feigenbaum *et al.*, 2014) a discrete spectrum of times at which an accountability mechanism might play a role. Although we categorize these points in terms of their goals or effects and not in terms of strict temporal relationships, there is a natural temporal ordering that applies.

Prevention: (At least partially) concerned with preventing violations; plays a role before a violation occurs

Violation: Plays a role at the time a violation occurs

Detection: Facilitates, enables, *etc.*, detection of a violation either at the time the violation occurs or afterward

Evidence: Helps gather or preserve evidence about a violation that may be used against the accused violator. Gathering and preservation may be (but are not required to be) connected to detection. In some settings, the evidence is intended for use by the system operator; in others, it is intended for presentation to a third party, *e.g.*, a judge in a court of law.

Judgment or blame: Renders a verdict about an actor's guilt or blameworthiness with respect to a violation. For example, the judgment might be a verdict in a court of law or a determination by a system administrator that a user violated system policy.

Punishment: Punishes a violator in some way. For example, a user who violated system policy may be banned from the system for a period of time.

A single accountability mechanism might be involved at multiple points on this spectrum.

We believe that punishment is the crucial element of accountability and have explored this idea formally in earlier work. One reason for this belief is that accountability definitions that focus on other points on our spectrum, *e.g.*, evidence or blame, often assume, implicitly or explicitly, that there is a punishing mechanism in their environments. While a definition might then say that an entity is "accountable" if there is evidence connecting it to its misdeeds or if there is a mechanism that will blame it for a policy violation that it commits, the assumption that there is a punishing mechanism means that evidence or blame alone is not really enough to achieve the intended effect. In some sense, the earlier points in our temporal spectrum are all building toward punishment.

We discuss the centrality of punishment and some other perspectives on accountability further in Sec. 2.2.8.

2.1.2 Information

One question about accountability is the extent to which it implicates privacy. We are concerned with the information learned about the violation, the violator, and actors who do not violate any policy. Relevant questions include:

Participant identity: Does the mechanism require system participants to have identities? If so, how broadly is an identity known (*e.g.*, is it only learned by a trusted third party, is it learned by a limited set of participants, or is it potentially learned by all participants)?

Violations: Are violations disclosed? If so, how broadly (with the same set of possible answers as for identity)? How soon after the violation is this information learned?

Violators: Is the violator identified as such? If so, how broadly is this identification made (with the same set of possible answers as for identity and violations)?

2.1.3 Action

We address the following questions about actions, both in general operation and in detection and punishment of policy violations.

Centralization (no violation): Is the accountability mechanism (as it operates in the absence of a detected violation) centralized or decentralized?

Centralization (violation): Does the mechanism respond to a violation (in the gathering of evidence, judgment, and punishment) in a centralized or decentralized way?

Punishment: If violators are punished, is punishment "automatic" or "mediated," as those terms are used in (Feigenbaum *et al.*, 2011)? If punishment is mediated, is the mediator a participant who might play other roles in the system, or is it a specialized entity?

Relies on violator participation: To what extent does the functioning of the accountability mechanism rely upon continued participation by, or access to, the violator? For example, is the violator only punished if he continues to interact with the system?

2.1.4 Other approaches to categorizing accountability

Outside of computer science, Bovens (2007), using his sense of accountability given in Def. 2.9 below, categorizes types of accountability based on four orthogonal dimensions: the nature of the forum to which an actor owes an account ("to whom is account to be rendered?"); the nature of the actor rendering account to the forum ("who should render account?"); the nature of the conduct ("about what is the account to be rendered?"); and the nature of the obligation that leads the actor to render an account to a forum ("why the actor feels compelled to render account"). Bovens illustrates each dimension with multiple distinct examples, *e.g.*, political, legal, administrative, professional, and social accountabilities that are distinguished by the nature of the forum to which an account is rendered.

Also outside of computer science, Lindberg (2013, pp. 212–217) synthesizes a categorization that contrasts with our approach above. His view of accountability, highlighted in Sec. 2.2.4, is that it captures a principal's delegating authority to an agent and then the agent's providing information for, and justification of, its decisions. Lindberg identifies three dimensions of accountability relationships: the *source*, *i.e.*, whether the principal is "internal or external to the one being held accountable[;]" the *degree of control*, which might be high or low; and the *direction* of control, which might be upwards (as with, *e.g.*, elected representatives), downwards (as with managers requesting information from their subordinates), or horizontal (as among professional peers). In addition to these examples, Lindberg provides examples for each of the 12 points in this space.

These categorizations are closely tied to the definitions of "accountability" used by Bovens and Lindberg. The examples illustrating them are potentially informative but not directly related to the approach that we take here.

2.2 Definitions of "accountability"

In surveying the literature, we identify numerous definitions, both explicit and implicit, of "accountability." Many definitions in the literature are essentially equivalent, and we do not present all of them. Rather, we categorize definitions based on their respective foci. Within each focus, we identify fundamentally distinct definitions and highlight some exemplars.

The first set of definitions focuses on detection of violations. The second set focuses on evidence of some sort, *e.g.*, the collection or presentation of evidence about participation or misbehavior. The third set is similar, but it focuses more tightly on identification of actors or the explicit association of actors with their actions. The fourth set addresses answerability for actions or explanation of actions to external parties; in some cases, it involves providing evidence, but we distinguish answerability from notions of evidence by focusing on frameworks in which the actors themselves provide the evidence (*e.g.*, that they acted properly). The fifth set of definitions focuses on blame for policy violations or misbehavior, and the sixth set focuses on punishment for such violations or misbehavior.

2.2.1 Detection

We first consider definitions in which the goal is to ensure that participants can detect policy violations. Although we classify them as "focused on detection," they are closely related to the production and collection of evidence.

In the temporal spectrum of Sec. 2.1.1, "detection" refers to evidence collected by a participant, and "evidence" refers to evidence that is intended to convince a third party.

In the context of accountable storage, Yumerefendi and Chase (2007, p. 2) present the following definitions:

Definition 2.1. A system is *accountable* if it provides a means to detect and expose misbehavior by its participants.

A system is *strongly accountable* if it provides a means for each participant to determine for itself [whether] others are behaving correctly,

without trusting assertions of misbehavior by another participant who may itself be compromised.

In addition to the detection aspect highlighted here, they frame their approach as allowing participants to "demonstrate that their actions are *semantically* correct, as well as properly authorized" (Yumerefendi and Chase, 2007, p. 28). Thus they take the complementary approaches of allowing participants to detect misbehavior by others and providing participants with evidence to demonstrate (to other participants) that they have behaved correctly.

In arguing that "information accountability" should become a primary means through which society addresses the question of appropriate use of sensitive information, Weitzner *et al.* (2008) offer this general definition:

Definition 2.2. *Information accountability* means that

- The *use* of information should be transparent so that it is possible to determine whether a particular use is *appropriate under a given [information policy]*, and
- The system enables individuals and institutions to be held accountable for [policy violations].

This definition treats transparency as the means of allowing determination of violations and (implicitly) punishment.[1] By contrast, the Yumerefendi–Chase definitions do not specify the means by which detection must happen.

In the strong Yumerefendi–Chase and the Weitzner *et al.* definitions, there is a requirement that it be possible "to determine" whether a violation has taken place. Both strongly suggest that evidence of some sort is required for this to be done and, implicitly, that it must be computationally feasible to make the required determinations.

[1] Orthogonal to our discussion here, this definition is also of interest because it shifts the focus from the disclosure of information to the use of information.

2.2.2 Evidence

Typically, definitions that focus on evidence highlight the need to convince a third party of something. In some but not all evidence-focused definitions, principals who have complied with system policy can convince the third party that they did not commit violations.

In defining a proof logic for accountability, Kailar (1996) views accountability as provable association:

Definition 2.3. Accountability is the property whereby the association of a unique originator with an object or action can be proved to a third party (*i.e.*, a party who is different from the originator and the prover).

Other definitions also focus on associating actions with principals, but this one explicitly requires provability to third parties. Some later work on logics for reasoning about accountability in protocols roughly follows the approach of Kailar. Kudo (1998) extends Kailar's logic to a "temporal accountability logic" and uses it to analyze a protocol for submitting something to a server where the submission must be made within a certain time window. Kungpisdan and Permpoontanalarp (2002) stress the ability to hide information from the third party, saying that accountability "involves the ability of a party, called a prover[,] to convince another party, called a verifier, that a statement is true *without revealing any secret information to the verifier*."

In describing an "accountable" timestamping service, Buldas, Lipmaa, and Schoenmakers (Buldas *et al.*, 2000b) focus on evidence that can be used to convince an external party that misbehavior has occurred (implicitly defining what it means to hold a participant accountable) and also on evidence that allows an honest participant to prove her innocence. In particular, they say that a timestamping service as a whole is "accountable" if (Buldas *et al.*, 2000b, p. 296)

> whenever there is at least one uncorrupted party (say, one honest client interested in legal value of a particular timestamp) during the creation of stamped information:
>
> **Fraud detection** The service makes the trusted third parties accountable for their actions by enabling a principal

> to detect and later prove to a judge any frauds affecting the relative ordering between timestamps.
>
> **Anti-framing** If a party has honestly followed the protocol but is still accused in forgeries, she can [disprove] any false accusations.

Bella and Paulson (2006) have taken accountability to mean the delivery to a principal of evidence, which can then be presented to a judge, of another principal's participation in a protocol. In general, their formal approach involves proving (1) the validity of the evidence and (2) some sort of fairness, *i.e.*, that one protocol participant gets evidence if and only if the other one does. Their general approach requires interpretation in the context of an individual protocol; we discuss their particular results in Sec. 4.2.1.

The PeerReview mechanism of Haeberlen, Kouznetsov, and Druschel (Haeberlen *et al.*, 2007) uses the following notion of accountable systems.

Definition 2.4. [A]n *accountable system* maintains a tamper-evident record that provides non-repudiable evidence of all nodes' actions.

Although this definition is stated in terms of evidence, it does not explicitly discuss the utility of that evidence in convincing third parties of claims. At the same time, it also differs in focus from a definition of Yumerefendi and Chase (2005) that it draws upon and that we review in Def. 2.5.

PeerReview, which we discuss in Sec. 3.2.2, has had significant influence on other accountability mechanisms. It is worth noting that Haeberlen *et al.* see the benefits of accountability as including deterrence, through the threat of punishment; fault detection, enabling "timely recovery of faulty nodes[;]" and providing evidence both to assign blame and to allow innocent principals to demonstrate their innocence.

Haeberlen (2010) argues for the importance of accountability in cloud settings and identifies factors that make it difficult to apply accountability mechanisms, such as PeerReview, to the cloud. His definition of "accountability" includes both association of actions with principals and evidence that can be used to convince a third party of a

claim; because of the latter feature, we include it here. In particular, (Haeberlen, 2010, p. 53) says that

> a distributed system is accountable if a) faults can be reliably detected, and b) each fault can be undeniably linked to at least one faulty node. More specifically, we consider systems that have the following features:
>
> **Identities:** Each action (such as the transmission of a message) is undeniably linked to the node that performed it;
>
> **Secure record:** The system maintains a record of past actions such that nodes cannot secretly omit, falsify, or tamper with its entries;
>
> **Auditing:** The record can be inspected for signs of faults; and
>
> **Evidence:** When an auditor detects a fault, it can obtain evidence of the fault that can be verified independently by a third party.

The comprehensive, tamper-evident record and the ability to audit it narrows the class of systems that would be considered accountable.

Haeberlen also postulates an Audit primitive that a customer could invoke to determine whether a cloud provider has fulfilled an agreement. Audit should either return an indication that the service did satisfy the agreement or evidence that it did not. Of particular interest here are his three goals for such a primitive (Haeberlen, 2010, p. 55):

> **Completeness:** If the agreement is violated, Audit will report this eventually, and it will produce evidence of the violation;
>
> **Accuracy:** If the agreement is not violated, Audit will not report a violation; and
>
> **Verifiability:** Any evidence of an alleged violation can be checked independently by a third party, even if the third party trusts neither the customer nor the provider.

He argues that the accuracy goal must be met in order to interest cloud providers (who would not want to be blamed falsely) but that the others could be strengthened or weakened. We view all three as conceptually important properties. An accountable system might or might not have them, but they should be considered explicitly in designing and analyzing accountable systems.

Of note, Haeberlen identifies particular aspects of privacy that he considers relevant to cloud accountability. He suggests that the cloud provider could keep a separate log for each of its customers in order to protect customer privacy. He also suggests that an Audit primitive might provide different information depending on who invokes it: A customer would learn less information, protecting the provider's internal workings, while the provider would learn more information to help troubleshoot its systems.

Although Haeberlen does not develop a particular system in this work, he does identify tools that might be used to provide accountability for cloud systems. These include tamper-evident logs, replay using virtual machines, and trustworthy timestamping.

2.2.3 Identification and association

The identification of the principal that performed a particular action and the general association of a principal with all of its actions have been prominent in computer-science definitions of "accountability" over many years. Definitions in this class vary with respect to the nature of "identification": Is it a nominal identity, *e.g.*, a username, that is associated with action(s) or a real-world individual who might have taken on multiple identites ("Sybils")?

In an early survey, the National Research Council (1991, p. 78) associates accountability policies with "knowing who has had access to information or resources." Lampson (2004) uses this association as the definition of "accountability" in an early treatment of security in terms of "locks" and "punishment." In describing after-the-fact defensive strategies, he also highlights recovery (or "undo[ing] the damage"), *i.e.*, restoring a system from backups. Although recovery is separate from accountability, it is worth noting that preventive and deterrent measures are not the only ones available.

Lampson also highlights auditing as a key component of practical security (along with authentication and authorization). In the context of a "guard" deciding whether or not a principal can access an object in a system, he says that

> [i]f the guard records the proof in a reasonably tamper-resistant log, an auditor can review it later to establish accountability or to determine whether the system granted some unintended access and why. Since detection and punishment are the primary instruments of practical security, this is extremely important (Lampson, 2004, p. 45).

Note that Lampson's view is consistent with our notions of detection, evidence, judgment, and punishment. We agree with him that auditing can play an important role in accountability but do not view it as inherently essential.

Early work of Ko, Frincke, Goan, Heberlein, Levitt, Mukherjee, and Wee (Ko *et al.*, 1993) looked at the problem of attackers' moving through a network, performing actions that might seem innocuous on a per-host basis but that would be identifiable as an attack when viewed at network scale. They identified "accountability" as a goal and took it to mean the ability "to associate all activities of multiple instances of the same individual to the same [. . .] identifier[.]"

Yumerefendi and Chase (2005) present the following view of accountability that has, directly and indirectly, influenced much subsequent work.

Definition 2.5. [A]n accountable system associates states and actions with identities, and provides primitives for actors to validate the states and actions of their peers, such that cheating or misbehavior become detectable, provable, and undeniable by the perpetrator.

They contrast this view with some earlier uses of "accountable," noting that "the goal is to assign responsibility for states and actions, and not just to track and audit access control decisions." Yumerefendi and Chase also present a high-level vision for accountable systems that fits with the approach subsequently implemented by Haeberlen *et al.* (2007) in PeerReview.

The Accountable Internet Protocol (AIP) of Andersen, Balakrishnan, Feamster, Koponen, Moon, and Shenker (Andersen *et al.*, 2008) takes accountability to be the "associatiat[ion of] an action with the responsible entity[.]" AIP's creators identify the detection and prevention of source-address spoofing and the stopping of unwanted traffic as applications of such an association. Taking this as a general definition requires explication of "responsible," which may be more complex in other settings.

The BackRef system of Backes, Clark, Kate, Simeonovski, and Druschel (Backes *et al.*, 2014) is designed to provide anonymous-communication networks with *repudiation* (allowing exit nodes to demonstrate that they did not originate a communication stream) and *traceability* (allowing the identification of the source of a stream when relays cooperate). This work does not provide a single definition of "accountability," but identification implicitly plays a major role in the way the concept is treated. Backes *et al.* note that other work on adding "accountability" to anonymous-communication networks has included "allowing misbehaving users to be selectively traced, exit nodes to deny originating traffic [they forward], misbehaving users to be banned, and misbehaving participants to be discovered."

Ishai, Ostrovsky, and Zikas (Ishai *et al.*, 2014) define a notion of *identifiable abort* for secure, multiparty computation. For an arbitrary functionality $\mathcal{F}$, they define a corresponding functionality $[\mathcal{F}]_{\perp}^{\mathtt{ID}}$ that

> behaves as $\mathcal{F}$ with the following modification: upon receiving from the simulator a special command (abort, p_i), where p_i [...] is a corrupted party [...], $[\mathcal{F}]_{\perp}^{\mathtt{ID}}$ sets the output of all (honest) parties to (abort, p_i). (Ishai *et al.*, 2014, p. 373)

In particular, when the protocol fails, all of the honest parties *agree on the identity* of a party that is indeed corrupt. Ihai *et al.* present results on multiparty-computation protocols in both information-theoretic and computational settings.[2]

[2] As noted by, *e.g.*, Ishai *et al.* (2014), while the classic work of Goldreich, Micali, and Wigderson (Goldreich *et al.*, 1987) predates the notion of "identifiable abort," the GMW protocol does indeed provide identifiable abort.

In providing an efficient approach to identifiable abort, Baum, Orsini, and Scholl (Baum *et al.*, 2016) consider how to ensure that the identity of the corrupt party that is identified to the honest participants can also be proved to outside parties. The ability to do this is not inherently part of identifiable abort. They also provide a concise review of the different lines of work on secure, multiparty computation with identifiable abort.

The works reviewed in this subsection illustrate what is in our view a fundamental gap in the computer-science literature on accountability. They present a broad, technically interesting variety of mechanisms for identifying entities *that have violated system policy and therefore should be held accountable for those violations*, but they do not give technical definitions of "held accountable." We view subsequent punishment as essential to the notion of holding violators accountable.

2.2.4 Answerability

The notion of accountability as *answerability*—*i.e.*, a requirement to give an answer for, or account of, actions and decisions—appears primarily in disciplines other than computer science. A comprehensive treatment of all such definitions is outside the scope of our survey, but we highlight some perspectives that take this viewpoint. While we argue for a view that focuses on punishment, we note that many of the answerability-focused perspectives implicitly assume a punishment mechanism, the existence of which implies that answerability produces deterrence.

Closely related to answerability is the notion of transparency. Instead of after-the-fact answering for actions and decisions, transparency opens up the decision-making process itself to inspection before, during, or after decisions are made. This variation on answerability has also been seen mainly in work outside of computer science.

First, some common English-language definitions point to this view of accountability. The Oxford English Dictionary defines *accountability* and *accountable* as follows:

Definition 2.6 (Accountability)**.** The quality of being accountable; liability to account for and answer for one's conduct, performance of duties,

etc. (in modern use often with regard to parliamentary, corporate, or financial liability to the public, shareholders, etc.); responsibility.[3]

Definition 2.7 (Accountable)**.** Chiefly of persons (in later use also organizations, etc.): liable to be called to account or to answer for responsibilities and conduct; required or expected to justify one's actions, decisions, etc.; answerable, responsible.[4]

In the context of summarizing and arguing against the expansion of "accountability" in public administration (see Sec. 3.6.2), Mulgan (2000) argues for the following.

Definition 2.8 (Mulgan's "core sense of accountability")**.** One sense of 'accountability', on which all are agreed, is that associated with the process of being called 'to account' to some authority for one's actions[.]

He argues that key components of the core sense are the fact that it is "external," that "it involves social interaction and exchange," and that "it implies rights of authority[.]" He also notes that sanctions may be implicit in authority but that the possibility of sanctions is less clearly required by "giving an account" than it is by "calling to account."

In related work on measuring accountability attributes in the cloud, Nuñez, Fernandez-Gago, Pearson, and Felici (Nuñez *et al.*, 2013) relate accountability to

> policy enforcement, risk management, incident management and remediation. Accountability is a high-level concept that entails all these practices. In general, accountability deals with being able to demonstrate that the accounts provided by an organisation (to regulators, auditors, data subjects or other service providers) are adequate and appropriate for the context, and implementing mechanisms for responding to the situation (including sanctions and remediation) if this is not the case.

They suggest "non-functional properties" of systems—"not directly related to functionality, but to a quality or behavioral attribute of a

[3] http://www.oed.com/view/Entry/1197, accessed August 11, 2017.
[4] http://www.oed.com/view/Entry/1198, accessed August 11, 2017.

system"—that are related to accountability. These include "transparency, verifiability, observability, liability, responsibility, attributability, and remediation."

Charlesworth and Pearson (2013) and Pearson and Wainwright (2013) both discuss accountability broadly, placing it in a social and legal context. They take an interdisciplinary approach to accountability, bringing together both legal and technical perspectives, with a focus on cloud-computing services. Pearson and Wainwright address overall security, reliability, and privacy for cloud computing services, while Charlesworth and Pearson focus specifically on privacy of personally identifiable information (PII). From a definitional perspective, Charlesworth and Pearson incorporate legal responsibility, which we view as a form of answerability, noting:

> [A]ccountability in this context [of Australian, U.S., and Canadian law] means placing a legal responsibility upon an organization that uses PII to ensure that contracted partners to whom it supplies the PII are compliant, wherever in the world they may be. Our accountability model reflects the basic premise of this approach, but extends it by suggesting ways in which organizations might take the "accountability" approach further in order to develop a reflexive, continually evolving, privacy process.

Pearson and Wainwright take a "co-design" approach, envisioning an accountability framework or ecosystem captured by a matrix. They place users, providers, and regulators on one axis and preventive, detective, and corrective mechanisms on the other. They also review various solutions developed in industry to address specific properties in the accountability matrix, namely risk assessment, data obfuscation, consent management, sticky policies, and monitoring for information use. Charlesworth and Pearson highlight privacy-related issues in cloud computing, including outsourcing, offshoring, virtualization, and autonomic technology. They discuss the extent to which regulation, such as the EU's privacy directive (EU Directive 95/46/EC) and the UK Data Protection Act of 1998, can and cannot handle such issues, and they suggest that an accountability-based approach that incorporates

co-design of legal and technical mechanisms and procedures would be more suitable.

In discussing different conceptions of accountability in information privacy and data protection, Raab (2012) highlights its connection to "stewardship, by which is meant that one party entrusts another with resources and/or responsibilities." He connects accountability in data protection to accountability in political sense. In particular, Raab notes (quoting Mulgan (2000)) that "[a]ccountability, in the sense of acquitting the responsibilities of stewardship through the giving of an account, is therefore somewhat similar to another meaning of accountability as 'dialogue', in which 'officials . . . answer, explain and justify, while those holding them to account engage in questioning, assessing and criticizing'."

We now note some additional perspectives from outside of computer science that fall under "answerability" and that have informed work in computer science. The differences among them highlight views that might be argued for or against when designing systems and mechanisms in computer science. However, we do not attempt to judge the relative merits of competing perspectives for use in other disciplines.

Writing about European policy, Bovens (2007) provides a self-described "narrow" sense of the concept of accountability in order to allow more concrete analysis than is allowed by broader uses of the term. For example, he notes:

> Particularly in American scholarly and political discourse, 'accountability' often is used interchangeably with 'good governance' or virtuous behaviour. (Bovens, 2007, p. 449)

As a short summary of his view, Bovens describes accountability as "the obligation to explain and justify conduct." Enriching this description, he makes the following definition:

Definition 2.9 (Bovens's "narrow" sense of accountability)**.** Accountability is a relationship between an actor and a forum, in which the actor has an obligation to explain and to justify his or her conduct, the forum can pose questions and pass judgement, and the actor may face consequences. (Bovens, 2007, p. 450)

Notably, in Bovens's view, the judgment that is passed is *on the conduct of the actor*.

Lindberg (2013) surveys conceptualizations of accountability across disciplines outside of computer science and categorizes types of accountability. He identifies in accounting, and political science, the "underlying principle of delegating some authority, evaluating performance, and applying sanctions" as a key aspect of accountability. In particular, Lindberg views accountability as distinct from responsiveness:

> At a very fundamental level then, accountability is closely associated with authority though not necessarily political authority. A puppet acting as an extension of someone else's will is not a legitimate object of accountability. That is why accountability is different from "responsiveness." Only actors with some discretion to make authoritative decisions can be the objects of accountability relationships[.] (Lindberg, 2013, p. 208)

Of note for our survey, Lindberg takes a different view from Bovens and focuses on sanctioning agents for the information and justifications they provide, not the underlying actions.

> [A]n important distinction should be made between the right to sanction [an agent] for failure to provide the information requested and justifications for decisions and actions taken, and the right to sanction agents or institutions [...] for the content or effects of such decisions and actions. At its core, accountability only necessitates the right to sanction [an agent] for failure to provide information and justify decisions. (Lindberg, 2013, p. 210)

As noted above, we do not take a position on which of these better comports with "accountability" as used in other disciplines. However, it is worth distinguishing between punishing or rewarding an agent for its underlying actions and for the information it provides about those actions.

Returning to computer science, Klonick (2018) provides an analysis of how and why online platforms (specifically Facebook, Twitter, and

YouTube) moderate content; she takes this analysis as a point of departure for a broader discussion of *online governance* and free speech (or the lack thereof) on private platforms, which she refers to as our "New Governors." Although Klonick does not define "accountability," she clearly uses the term to mean answerability. She describes the platforms as "private, self-regulating entities" that are inclined to support the free-speech expectations of their users because it is good for their bottom lines to do so and because free speech is a cherished value in the tech world; rules and procedures for moderating content are heavily influenced by US free-speech norms. Klonick identifies the main threats to democratic culture posed by private online governance as potential loss of fair opportunity to participate (given that platform operators can block users whose posts violate their "community standards") and platforms' "lack of direct accountability to [their] users." Attempts to counter these threats, she argues, should start with technical changes to the architecture and content-moderation systems put in place by platforms. If a purely technical approach fails, she advocates regulation that emphasizes balance between the democratizing force of the internet (which requires fair opportunity to participate) and the innovative force of the New Governors.

Kroll (2015)—in his thesis focusing on "a *decision authority* making decisions using a set *policy*[,]"—defines "accountable" as follows. He notes that it requires not just explainability but also consistency with norms.

Definition 2.10 (Def. 2 of (Kroll, 2015))**.** We say that a process or entity is *accountable* for a decision if that process is consistent with the *legal, political, and social norms* that define its context, and this fact is publicly evident, either because such consistency can be readily determined using public information or because an entity empowered and trusted to define such consistency (*e.g.*, a court, a legislature, or other designated authority) can be shown to have the necessary information to determine this consistency.

He identifies four properties as necessary for a decision process to be accountable (Kroll, 2015, p. 57):

1. the authority must be committed to its policy in advance;
2. the result asserted by the authority must be the correct output of the authority's committed policy when applied to an individual decision subject;
3. any randomness used in the decision policy must be generated fairly; and
4. if necessary, the authority can reveal its policy to an oversight body for examination later, without revealing decision subjects' private data, and that body as well as each decision subject can verify that the revealed policy is the one that was used to make the decisions in question.

In Kroll's view, accountability is "inherently political," and it may thus be impossible to determine sufficient properties to ensure that a decision process is accountable. This perspective suggests that transparency may be more important than some other answerability-focused perspectives, but subsequent work by Kroll, Huey, Barocas, Felten, Reidenberg, Robinson, and Yu (Kroll *et al.*, 2017) identifies the limitations of an approach that leans too heavily on transparency; see Sec. 2.3.2.

2.2.5 Blame

Accountability approaches that focus on blame are in some ways similar to those that focus on associating principals with the actions they perform. Indeed, such associations can be useful in assigning blame or responsibility. However, we distinguish these two types of approaches. We regard blame or responsibility (terms we use interchangeably) as something that requires additional analysis beyond the mapping of actions to principals. The concept of blame captures the possibility that, although action a done by principal P violates a system policy, actions previously taken (or neglected) by principal Q mean that *no* action done by P could possibly satisfy system policy—nor could the policy be satisfied by the complete *lack* of action by P. Alternatively, Q could

have coerced P into doing a. Blaming Q for a system violation in such cases requires more than simply knowing that it was P who did a.

The 1991 National Research Council (NRC) report provides the following definitions that are relevant for historical context.

Definition 2.11.
Accountability The concept that individual subjects can be held responsible for actions that occur within a system.
Auditing The process of making and keeping the records necessary to support accountability.
(National Research Council, 1991, pp. 286–287)

By contrast, the report notes elsewhere that "individual accountability" has classically been part of management controls:

> *Individual accountability* answers the question: Who is responsible for this statement or action? Its purpose is to keep track of what has happened, of who has had access to information and resources and what actions have been taken. In any real system there are many reasons why actual operation may not always reflect the original intentions of the owners: people make mistakes, the system has errors, the system is vulnerable to certain attacks, the broad policy was not translated correctly into detailed specifications, the owners changed their minds, and so on. When things go wrong, it is necessary to know what has happened, and who is the cause. This information is the basis for assessing damage, recovering lost information, evaluating vulnerabilities, and initiating compensating actions, such as legal prosecution, outside the computer system. (National Research Council, 1991, p. 57)

Finally, the NRC connects accountability policies to "knowing who has had access to information or resources" (National Research Council, 1991, p. 78); this is informed by the perspective of Lampson discussed in Sec. 2.2.3.

Irwin, Yu, and Winsborough (Irwin *et al.*, 2006) present a framework for modeling and analyzing *positive* obligations, *i.e.*, requirements that

particular actions be taken during a particular time window. Obligations might be imposed as a condition of allowing certain actions to be performed, and they complement preventive access-control measures. Noting that system participants typically cannot be compelled to satisfy their obligations, Irwin *et al.* seek systems in which it is possible to fulfill all obligations. In such systems, unfulfilled obligations represent violations by participants (instead of, *e.g.*, tasks made impossible by scheduling or resource restrictions), and the goal is to identify the participant that did not fulfill its obligations. Informally, Irwin *et al.* say that "[a] system is accountable if and only if all the obligations will be fulfilled supposing all the subjects are diligent. In other words, a secure state implies that any obligation violation can only be due to the lack of diligence of subjects." They then study the feasibility of checking whether systems are accountable in this sense; we discuss this and subsequent work in Sec. 4.3.1.

Irwin *et al.* also say:

> Rather than requiring that it be impossible for obligations to be violated, instead we assume that it is possible that obligations go unfulfilled, but when they do, we would like to clearly identify whose fault it is. Obviously, an obligation can go unfulfilled because a subject simply fails to take the required action before the deadline, even if he has sufficient privileges and resources. It is desirable that this is the only reason that an obligation will go unfulfilled.
>
> Intuitively, if it is the case that all users have sufficient privileges and resource to carry out their obligations so long as every other user carries out his or her obligation, then a system is said to be in an *accountable state*, because we can know that whoever first fails to carry out an obligation is responsible for the violation and anything which results from it.

Datta *et al.* (2015) distinguish accountability from punishment ("determining which agents should be held accountable and appropriately punished is important to deter agents from committing future violations") and identify assigning blame to an agent in a system with holding

agents accountable. This is expanded upon by Sharma (2015) in her dissertation, where she also provides an overview of other accountability-related work. We note that the view of Datta *et al.* does not encompass our notion of automatic punishment.

In connecting this perspective to technical, computer-science definitions, Datta *et al.* (2015, Sec. VI) argue that

> accountability is not a trace property since evidence from the log alone does not provide a justifiable basis to determine accountable parties. Actual causation is not a trace property; inferring actions which are actual causes of a violating trace requires analyzing counterfactual traces[. . . .] Accountability depends on actual causation and is, therefore, also not a trace property.

Kacianka, Beckers, Kelbert, and Kumari (Kacianka *et al.*, 2017) survey the literature on implementations of accountability mechanisms and techniques. One of their goals is to identify a common definition of "accountability." While they find no such common definition in the literature they survey, they offer the following four elements of a "work-in-progress definition of accountability" that focuses on blaming entities for actions:

1. *Accountability* is a property of a system or a collection of systems and is ensured by an Accountability Mechanism.

2. An *Accountability Mechanism* is part of an *Accountable System* and reasons over a tamper-proof log to link effects of that system to entities.

3. An entity is (partially) *accountable* for a given effect if an *Accountability Mechanism* can prove a causal link between the entity's action and the given effect.

4. The set of entities *accountable* for a given effect is the set of all entities for which an *Accountability Mechanism* can prove a causal link between the entities' actions and the given effect.

2.2.6 Punishment and deterrence

In Sec. 2.2.3, we pointed out that some key definitions of "accountability" that focus on association between principals and actions tacitly assume the existence of a punishment mechanisms in the background. By contrast, the perspectives we review in this subsection make punishment itself an explicit focus of the definition.

Lampson (2005a) takes an early step toward the focus on punishment and deterrence:

Definition 2.12. Accountability is the ability to hold an entity, such as a person or organization, responsible for its actions.

He notes that "[a]ccountability is about punishment, not locks" and, in the context of an anti-spam policy, that "[s]enders must be able to make themselves accountable: This means pledging something of value." In general, he identifies deterrence as the goal and punishment as a tool to achieve it, thereby conceptualizing online enforcement of system policies as analogous to real-world law enforcement. He later draws this connection explicitly in (Lampson, 2009, p. 27), where he says that "real-world security depends mainly on deterrence, and hence on the possibility of punishment."

One can view this formulation as an evolution of the NRC definition used earlier in (Lampson, 2004). The focus shifts from "knowledge about access" to deterrence via punishment, but punishment may be *enabled through* knowledge about access. Indeed, Lampson (2005a) envisions using "a consistent identifier based upon a name, a pseudonym, or a set of attributes" to punish a principal by tarnishing its reputation. This illustrates the often close relationship between identification of principals with actions and punishment or deterrence.

We also recall the definition of Grant and Keohane (2005) that we noted in Sec. 1.2.1:

Definition 2.13. Accountability, as we use the term, implies that some actors have the right to hold other actors to a set of standards, to judge whether they have fulfilled their responsibilities in light of these standards, and to impose sanctions if they determine that these responsibilities have not been met.

Grant and Keohane's definition is similar in some ways to Def. 2.12, but it even more clearly implies that the punishing agent(s) play an active role. It also explicitly encompasses more of the temporal spectrum of Sec. 2.1 than simply punishment.

In (Feigenbaum *et al.*, 2011), we offered a preliminary definition that avoids the assumption made in Defs. 2.12 and 2.13 that there is an active entity that holds the violator responsible for his actions by punishing him. Our formulation instead allows for the possibility that there is no such entity and that punishment occurs automatically:

Definition 2.14 (Working definition of "accountable entity")**.** An entity is accountable with respect to some policy (or accountable for obeying the policy) if, whenever the entity violates the policy, then with some non-zero probability it is, or could be, punished.

The inclusion of "or could be" was intended to address cases in which the accountability mechanism *could* punish a violator but chooses not to in order to serve another system goal. In real-world law enforcement, an analogous notion allows for the possibility that a criminal may be caught but not prosecuted in order to avoid revealing the existence of an ongoing investigation into more serious crimes.

Definition 2.14 was preliminary and, in retrospect, flawed. For example, the "non-zero probability" requirement is too weak. However, we maintain that it is important not to require that punishment happen with absolute certainty for a number of reasons, *e.g.*, the law-enforcement scenario described above. Datta has pointed out a more significant issue with Def. 2.14, namely that it would be satisfied by punishing *all system participants* whenever a violation is committed; such collective punishment violates widespread social and ethical norms (including those articulated in the Geneva Conventions (Uhler *et al.*, 1958, Article 33)) and would be undesirable in typical systems. We subsequently considered (Feigenbaum *et al.*, 2014) the requirement that punishment be "targeted" at a blameworthy principal.

Deterrence is also considered in cryptography. For example, Aumann and Lindell (2010) define the notion of *covert adversaries*, which lies between the notions of arbitrarily malicious adversaries and honest-but-curious adversaries. These adversaries are "willing to actively cheat [. . .],

but only if they are not caught[.]" An attempt to violate the protocol is detected with probability ϵ, which is called the "deterrence factor."

2.2.7 Other senses

Clark (1988) reviews the design philosophy of the DARPA Internet protocols. His review has been cited by supporters of the argument that "accountability" was always an Internet design goal; therefore, it is important to understand the context for his statement and the sense in which he used the word "accountable."

Clark gives an ordered list of seven second-level design goals, the least important of which is that "resources used in the Internet architecture must be accountable." Implicitly recalling the DARPA context, he goes on to explain that "a detailed accounting of resources used" is less important in wartime, but he notes that "non-military consumers . . . are seriously concerned with understanding and monitoring the usage of the resources within the Internet."

He also points to the earlier work of Cerf and Kahn, in which "accounting" is identified "as an important function of the protocols and gateways." Fortunately, "accounting" is not as hard a term to pin down as "accountability," and the authors take it to mean what it usually means in a business context (Cerf and Kahn, 1974, p. 638):

> To allow networks under different ownership to interconnect, some accounting will undoubtedly be needed for traffic that passes across the interface. In its simplest terms, this involves an accounting of packets handled by each net for which charges are passed from net to net until the buck finally stops at the user or his representative.

Cerf and Kahn suggest that "account [be] taken" of units of flow at gateways between networks, and they note possible accounting effects of fragmentation.

These early works focused primarily on costs. However, like many more recent works that focus on security, they emphasize association between actions and principals.

In writing about accountability in public administration, Koppell (2005) quotes

> [t]he *Public Administration Dictionary*[, which] defines accountability as "a condition in which individuals who exercise power are constrained by external means and by internal norms[.]"

This broad view cuts across the mostly computer-science derived categories that we have identified in this section.

2.2.8 Terminology: "accountability" *vs.* "deterrence"

Among the numerous computer-science definitions of "accountability" that we have presented, few focus primarily on punishment, but many assume—implicitly or explicitly—the existence of a punishment mechanism. We take this as support for our view that punishment is key to many views of accountability.

Although our initial formulation in Def. 2.14 focuses on punishment, there are good arguments for the view that "accountability" necessarily involves other properties. Indeed, it makes sense to regard punishment alone as the essential foundation of "deterrence" but not of "accountability." Weitzner (2017), for example, stresses the interactive, social, and educational role that accountability mechanisms typically play in communities. He observes that, when a convicted criminal is "held accountable" for his crime, there is more to the process than imposition of an appropriate punishment. Importantly, the judge announces the sentence in an open court (in the presence of the prosecutor and the defense attorney, often along with the victim and her supporters, interested members of the public, and the media) and may explain why the sentence is on the high or low end of what is typical for this crime. In this way, the criminal, the victim, and other key members of the community receive an "account" through which they can understand not only that a crime has been committed and punished but the nature of the crime and why the punishment fits it.

By contrast, our notion of "automatic punishment" allows a policy violation to be punished without the system participants or even the violator himself learning than a violation was committed or by whom. This process may provide deterrence if the policy and the punishment

terms are well publicized, but it does not necessarily provide the system participants with understanding or social recognition.

We view this terminological question as important for facilitating communication but separate from the identification of key properties in this space. Regardless of terminology, punishment is a key component of many of the approaches to information security that are not purely preventive.

2.3 Accountability-related concepts and terms

We now describe approaches to the relationships among different terms surrounding accountability. We start by updating the perspective that we presented in some earlier work (Feigenbaum *et al.*, 2014). We then briefly consider "accountability" in relation to "transparency," "causality," and "blameworthiness."

2.3.1 Principals, nyms, actions, and their relationships

This work considers abstract systems, examples of which include multi-user computers, computer networks, and offline settings such as open-air markets. Entities participate in the system as follows. An individual, called a *principal*, takes on an identity in the system, called a *nym*, and acts within the system as that nym. We allow for the possibilities that a principal uses more than one nym and that one nym might be used by more than one principal. We connect principals, nyms, and actions, as well as the directed relationships between them, to various accountability-related concepts, as illustrated in Figure 2.1. We incorporate and update the perspective that we offered in (Feigenbaum *et al.*, 2014).

Figure 2.1 includes a principal, a nym, and an *action*. The dashed line surrounding the nym and action indicate that they are "inside" the *system*, while the principal is "outside" the system. The nym might use the same identifier as the principal. For example, a person (the principal) might be required to use her real name (a nym) to open a bank account and make a deposit (an action) within the banking infrastructure (the system). Although the nym matches her name, the

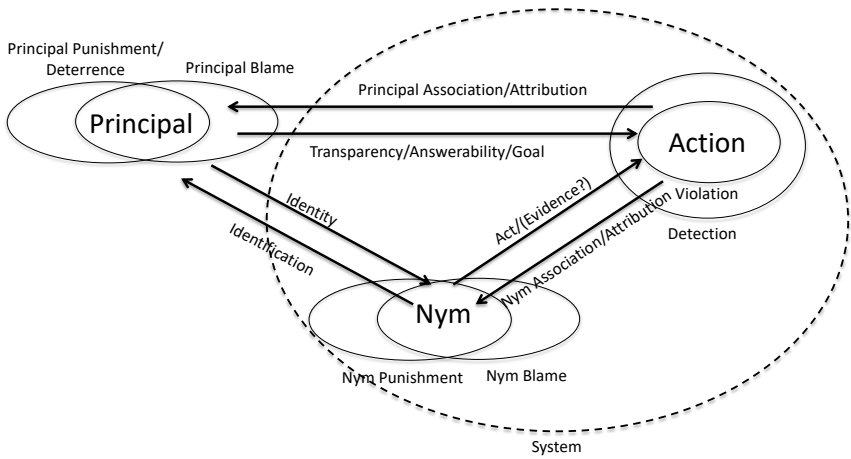

Figure 2.1: Concepts mapped to relationships among principals, nyms, and actions.

individual exists outside the system, and her matching nym exists inside the system. The label "*act*" on the arrow from the nym to the action indicates that it is the nym (not the individual human) that acts in the system.

We use *identity* to describe a principal's taking on a nym, and we use *act* to capture a nym's acting in the system. These terms appear on the directed edges from principal to nym and from nym to action, respectively, in Fig. 2.1. We use an oval labeled "*violation*" around an action to indicate that the action violates system policy. The label "*identification*" indicates a determination of which principal was using a nym (*e.g.*, at a particular time or to do a particular action). Because this starts with the nym, we put this term on the edge from the nym to the principal.

Categories of accountability definitions given in Sec. 2.2 appear in Fig. 2.1 as well. *Detection* of a violation appears as a larger oval around

the "violation" oval. (This does not guarantee that all violations are detected.) *Evidence* that can be presented to a third party might take a variety of forms, but it often involves (*e.g.*, as in PeerReview) the connection between nyms and their actions. We thus put "evidence" on the edge from the nym to the action; the question mark indicates that this is not the only type of evidence that might be collected. Actions might be associated with either nyms or principals; thus, we identify both *nym association* and *principal association* and connect them to the edges from the action to the nym and the principal, respectively. A similar concept is that of *attribution*, again with both nym and principal varieties. Most of the directed relationships (*i.e.*, not violation or detection) considered so far have complements obtained by reversing the directions of arrows. The exceptions are principal association/attribution, which maps an action to a principal.

Blame may be attached to both nyms and principals, and Fig. 2.1 depicts these types of blame as ovals around the nym and the principal, respectively. As with association/attribution, blame of a principal might be achieved in practice by blaming a nym and then (correctly) identifying the relevant principal.

Finally, Fig. 2.1 depicts *punishment* of both principals and nyms. Punishing a principal might be effected by "punishing" a nym if doing so subsequently punishes the principal. For example, decreasing a nym's reputation might punish the sole principal who can use that nym by making the nym less useful. On the other hand, a nym is not actually punishable in the same way that a principal—*e.g.*, a human individual—is. To clarify this point, we add "*deterrence*" to the oval labeled "principal punishment" in Fig. 2.1, but we do not do so for the nym. Principals might be deterred, but nyms, which lack the human elements of intention, will, or desire, cannot be.

2.3.2 Transparency and accountable algorithms

"Accountability" for algorithms has been a topic of increasing interest. This interest has been fueled, at least in part, by results showing that algorithms can encode bias even if their designers wanted to avoid it (Barocas and Selbst, 2016). The FAT/ML (Fairness, Accountability,

and Transparency in Machine Learning) effort[5] has articulated principles for accountable algorithms (Diakopoulos *et al.*, n.d.). In the context of automated decision making, those principles describe accountability as "includ[ing] an obligation to report, explain, or justify algorithmic decision-making as well as mitigate any negative social impacts or potential harms." More recently, the ACM FAccT conference[6] (ACM Conference on Fairness, Accountability, and Transparency) has grown out of FAT/ML and similar organizations in adjacent specialties. Among the related concepts addressed by this research community, one that is often treated as clearly important is *transparency*; both technologists and the general public express support for the idea that the intent and consequences of widely used algorithms should be easily accessible. Developers who open source their code, for example, often claim that they are doing so in order to achieve transparency and accountability.

In recent interdisciplinary work, Kroll *et al.* (2017) argue against transparency as a standard, saying that

> [d]isclosure of source code is often neither necessary (because of alternative techniques from computer science) nor sufficient (because of the issues analyzing code) to demonstrate the fairness of a process. Furthermore, transparency may be undesirable, such as when it discloses private information or permits tax cheats or terrorists to game the systems determining audits or security screening.

They instead argue for "procedural regularity, meaning that decisions are made under an announced set of rules consistently applied in each case" (Kroll *et al.*, 2017, p. 634). They also look for technical ways to "assure that automated decisions preserve fidelity to substantive legal and policy choices." This line of work forms one point of departure for the study of "accountable algorithms."

[5] https://www.fatml.org/, accessed November 12, 2020.
[6] https://facctconference.org/, accessed November 12, 2020.

2.3.3 Relationship of accountability to blameworthiness and causality

In Sec. 2.2.5, we discussed definitions of accountability that focus on blame. Stepping back from the details of precise definitions, one could take a common-sense approach the goal of which is to "hold accountable" the entity that is "to blame for" or that "caused" a violation of some policy. Specifying an accountability mechanism that achieves this goal requires an understanding of blameworthiness or causality as well as accountability. Both blameworthiness and causality have been the subjects of extensive study in computer science, a full review of which is beyond the scope of our survey. Here we briefly discuss several works that treat blame or causality in ways that are particularly relevant to accountability in computer systems.

In our own updated formalization of "mediated" punishment (Feigenbaum *et al.*, 2014), we require that a punishing action be causally dependent upon the fact that a principal is blameworthy. This is to ensure that bad luck does not satisfy the definition of punishment. For example, a bank robber should not be considered to have been punished if she gets hit by a bus the day after robbing a bank. However, in our formalizations of accountability, we abstracted away the process for determining causality.

Barth *et al.* (2007) present a logic for reasoning about business processes and determining whether they achieve privacy and organizational goals. Their work includes algorithms for identifying agents who are potentially to blame for policy violations, but they note that, according to their definitions, not every participant who *might* be blamed actually caused a policy violation.

In particular, they consider whether, in the view of any participant, an action i occurred before an action j. The minimal transitive relation capturing this property is the *possibly-caused* relation; it captures their standard, trace-based notion (Lamport, 1978) of all of the possible causes of the action j. They note that, if i caused j in another view of actual causality, then i is captured as a possible cause of j but that the converse need not hold. They then define *accountable* participants as follows:

Definition 2.15 (Accountable agents in (Barth *et al.*, 2007))**.** An agent is *accountable* for policy violation i [...] if the agent undertook an action [in the possible causes of i] and did not fulfill his or her responsibilities[.]

Under this definition, every agent whose irresponsibility actually caused a violation is classified as accountable, but not every accountable agent actually caused a policy violation.

Datta *et al.* (2015) consider program behavior as actual causes as part of determining blameworthiness for a violation. They highlight important differences between their work and other approaches. In particular, they distinguish between deviations from prescribed behavior and causes of violations and suggest that the punishment for the two might appropriately differ. For example, *causing* a car crash might be punished more severely than driving in a reckless fashion (risking but not causing a car crash). Datta *et al.* also identify *sequences of actions* as violations and not just single events. We discuss their approach in more detail in Sec. 4.1.3.

Notes

Accountability has been considered in many other works, most of which overlap with the ones covered in this chapter. We briefly discuss some of them here.

In the context of content-distribution architectures, Ó Coileáin and O'Mahony (2015, p. 59:2) distinguish between *accounting*, which focuses on business needs, and *accountability*, which focuses on using evidence to identify a principal that has committed a violation.

Three European projects considered accountability in the context of online privacy and data-protection regulation. The EC Article 29 Data Protection Working Party added a "principle on accountability" to the EU Data Protection Directive, stating that data controllers must take appropriate and effective measures to implement data protection, must demonstrate upon request that such measures had been taken, and would be subjected to sanctions if they failed to fulfill this principle (EU Article 29 Working Party, 2010). The Cloud Accountability Project[7]

[7] https://a4cloud.eu/, accessed November 10, 2020.

identified the need for cloud-service providers to accept responsibilities, to explain them and demonstrate compliance to stakeholders, and to remedy failures to comply. In their review of this project, Felici *et al.* (2013) refined the definition of accountability and connected it to the notions of "data stewardship" and "data governance" in the cloud. The Accountability Project, often referred to as The Galway Project (Centre for Information Policy Leadership, 2009), also developed the notion of accountability as the assumption of responsibility for data protection, emphasizing the need for systems and mechanisms that would enable external verification, policy enforcement, and remediation for failure; in particular, Galway identified the need for accountability in cross-border transfers of data.

Dingledine, Freedman, and Molnar (Dingledine *et al.*, 2001) give a discussion of "accountability" in the context of peer-to-peer (P2P) technologies. They emphasize micropayments (though not necessarily involving the exchange of money or even computer resources) and reputation. One argument that emerges in their discussion is that, in systems with no identity, there can be no reputation, which means that payments need to be used. They also note that

> Identity does not imply accountability. For example, if a misbehaving user is in a completely different jurisdiction, other users may know exactly who he or she is and yet be unable to do anything about it.

This fits with a punishment-centric view of accountability, although Dingledine *et al.* also touch on prevention, detection, and identification without giving a formal definition of "accountability."

Wieringa (2020) has systematized existing work on accountable algorithms, using Bovens's perspective to categorize papers. Kroll (2020) broadens the scope out from algorithms specifically and provides a survey of accountability in computer systems.

Sullivan *et al.* (2010) present a trust-terms ontology that captures relationships among different terms and concepts surrounding trust and security. In relating the terms they consider, they discuss the relationships "requires," "facilitates," "implies," and "fosters." Although

they do not give a formal definition of "accountability," they do say that it fosters trust, facilitiates responsibility, and requires auditing.

Halpern and Pearl have developed one of the most prominent approaches to causality in computer science (Halpern and Pearl, 2005a; Halpern and Pearl, 2005b; Halpern, 2015; Halpern, 2016; Pearl, 2000). Their line of work is based on counterfactuals and on equational models that capture relationships between different events in a system.

Gössler and Le Métayer (2015) propose a trace-based definition of causality for violations of safety properties. Their basic apporach is to take observed logs of system behavior when there is a violation, identify behaviors consistent with those logs, change some of those behaviors to be correct, and then consider the logs that would correspond to the corrected behaviors. Gössler and Le Métayer argue that this approach may be better suited to some computing systems than the equational models of the Halpern–Pearl approach.

Finally, the PhD dissertations of Kroll (2015) and Sharma (2015) develop in full many aspects of accountability and blameworthiness that are touched upon in this chapter.

3

Accountability Mechanisms and Domains across Disciplines

In this chapter, we survey a range of accountability mechanisms and systems that use them, with a focus primarily on domains within computer science. Following the post-violation parts of the temporal spectrum that we presented in Sec. 2.1.1, we categorize our discussion of mechanisms and systems into those that focus on evidence (Sec. 3.1 and Sec. 3.2), judgment or blame (Sec. 3.3), and punishment (Sec. 3.4). Within these areas, rather than attempting to include every system and mechanism described in the literature, we instead identify exemplars of the most significant approaches. In practice, mechanisms that detect violations typically produce some sort of evidence, so we do not highlight a separate class of detection-focused mechanisms in this chapter.

For each system or mechanism highlighted in Secs. 3.1–3.4, we discuss: its goals and approach; where on our temporal spectrum of Sec. 2.1.1 the system or mechanism fits; the level of identity required to participate in the system or mechanism; how broadly violations are disclosed, and how broadly violators are identified as such; whether the system or mechanism is centralized or decentralized, with and without a violation occurring; whether the punishing entity, if applicable, is internal to the system/mechanism or whether it is external (*e.g.*, the

legal system); and whether any punishment for violations requires ongoing involvement in the system/mechanism by the violator. The classifications discussed in Sec. 3.1–3.4 are summarized in tables in Sec. 3.5.

We with a discussion (Sec. 3.6) of some additional accountability mechanisms in disciplines beyond computer science and then some concluding notes.

3.1 Evidence without focus on external parties

Accountability mechanisms and systems that focus on evidence might be viewed from different perspectives. We choose to classify these mechanisms according to whether or not the evidence they provide is intended to be useful to a third party, such as a court of law. In this section, we discuss mechanisms and systems that provide evidence mainly to a wronged party. This evidence would enable the wronged party to take actions such as ceasing to providing service or conducting rigorous further investigation. In Sec. 3.2, we discuss mechanisms and systems that provide evidence that is intended to be given to a third party.

An alternative perspective might classify evidence-focused mechanisms according to the technical approaches they take. That approach might contrast, *e.g.*, mechanisms that leverage cryptography to bind identities to actions with mechanisms that use lighter-weight approaches like polling a network without using cryptographic signatures. Mechanisms that provide evidence for third parties typically uses cryptography, but some that provide evidence mainly to the wronged parties also use cryptography.

3.1.1 Assurance about the source of network traffic

Accountable Internet Protocol (AIP)

The Accountable Internet Protocol (AIP) of Andersen *et al.* (2008) provides its notion of accountability—associating actions to identities—through the use of self-certifying addresses. As Andersen *et al.* describe it, this means that "the name [(*e.g.*, the host ID)] of an object is [...] the hash of the public key [...] that corresponds to that object" (*e.g.*,

the hash of the host's public key). They argue for this approach as follows (Andersen *et al.*, 2008, p. 340):

> Our use of self-certifying addresses follows from a simple line of reasoning. Accountability requires a verifiable identity, and in a network setting the only practical method of verification uses cryptographic signatures. To use such signatures, identifiers must be bound to their public key. Security, however, should not rely on extensive manual configuration or globally trusted authorities, so the keys must be intrinsic to the identifiers. Thus, we believe that self-certification is an indispensable aspect of providing accountability at the network layer.

Andersen *et al.* envision using AIP's accountability property to detect packets that use spoofed source addresses, forwarding only packets with correct addresses, and (building on earlier suggestions) stopping unwanted traffic based on a request from the recipient of such traffic. They also note that the AIP infrastructure would facilitate securing interdomain routing, but that benefit does not derive from the accountability property itself.

We may thus view AIP as providing some **detection** and **prevention** (of address spoofing) in order to provide the claimed accountability property. That, in turn, is used to facilitate either **punishment** or **prevention** of additional harm, depending on how one views the stopping of unwanted traffic.

By virtue of self-certifying addresses, AIP requires **identities** in the form of public keys for network hosts. These are not linked to individuals who might use those hosts, however. **Violations** are disclosed and violators are **identified** as such to network participants (when the recipient of unwanted traffic requests that further traffic from that violating source be stopped). Regardless of whether there is a violation, AIP is **decentralized**. To the extent that there is **punishment**, it is done internally by network participants who implement AIP's "shut-off protocol." This punishment or prevention, depending on one's perspective, requires ongoing **involvement** by the violator. If the violator leaves the system, it achieves the same effect.

AIP explicitly avoids the need for a public-key infrastructure (PKI) to associate keys to principals. However, it does make use of asymmetric cryptography to verify that traffic comes from the claimed address (which is the hash of a public key).

We note that self-certifying addresses are part of Tor's onion-site infrastructure (Syverson and Boyce, 2016). Self-certification is an inherent part of that infrastructure and not done specifically to achieve an accountability-related property, but it does provide an example of real-world deployment.

Accountable and Private Internet Protocol (APIP)

Naylor, Mukerjee, and Steenkiste (Naylor *et al.*, 2014) build on the approach of AIP but seek to balance accountability and privacy in their development of the Accountable and Private Internet Protocol (APIP). APIP introduces a separate "accountability address" to decouple identity from accountability. Naylor *et al.* (2014, p. 77) identify "accountability" as follows:

> At the network layer, by accountability we mean that *hosts cannot send traffic with impunity: malicious behavior can be stopped and perpetrators can be punished*. Specifically, we would like our design to have the following three properties:
>
> 1. Anyone can verify that a packet is "vouched for"—someone is willing to take responsibility if the packet is malicious.
> 2. Malicious flows can be stopped quickly.
> 3. Future misbehavior from malicious hosts can be prevented (*i.e.*, by administrative or legal action).

Here we see a number of different accountability-related properties combined. APIP achieves these goals by using trusted "accountability delegates." A packet sender's accountability delegate is identified by the accountability address included in traffic, which is separate from the return address that the recipient uses for replying (although this might be obscured from the destination via Network Address Translation).

Routers that receive a traffic flow can ask the named accountability delegate whether it vouches for the packets. (Naylor *et al.* contrast this with AIP, in which they say a "challenge asks, 'Is this packet's source address spoofed?'") Recipients of malicious flows can then ask the delegate to shutoff—by ceasing to vouch for—the bad flow. Longer-term fixes are envisioned through additional, external mechanisms.

When an accountability delegate stops verifying a sender's traffic, this helps other nodes in the network **detect** misbehavior. APIP provides some measure of **prevention** after initial bad behavior is detected; as with AIP, this might also be viewed as **punishment** in the sense that stopping malicious flows punishes the sender.

APIP requires the sender and accountability delegate to share a key, which it presumes is generated when they establish a contractual relationship. We take this as an **identity** requirement on the sender, but only revealing this to the delegate instead of more broadly. Any node in the network that receives the flow can query the delegate about it and learn that it was no longer being vouched for, so the fact of the violation is disclosed **broadly**. However, the identity of the sender who committed the **violation** may remain unknown except to the accountability delegate. The accountability delegates are largely **decentralized**, although in practice they may be more centralized, both with and without a violation. The **punishment** that does occur is meted out within the system, but it assumes continued participation (else ceasing to vouch for traffic, and subsequently dropping it, is not a punishment).

Pretty Good Packet Authentication

Haeberlen, Rodrigues, Gummadi, and Druschel (Haeberlen *et al.*, 2008) "settle for a narrower goal" than APIP. They describe Pretty Good Packet Authentication (PGPA), which achieves the following property:

> Given a packet P, a timestamp t *that is not more than* $T_{\max}$ *in the past*, and a source IP address S, the ISP that owns S can verify whether S has sent P at approximately time t, provided that monitors are deployed on that ISP's access links.

This approach provides evidence of the guilt of addresses that did send objectionable packets as claimed, and it provides evidence of the innocence of addresses that did not. PGPA works by attaching a monitor to the customer–ISP link; the monitor computes and stores packet hashes and timestamps. If the ISP is presented with a policy-violating packet, it can query the monitor to determine whether that packet was sent across that link.

Haeberlen *et al.* discuss privacy implications of the approach used by PGPA. Accusing an address requires possession of a packet and timestamp, which defends against arbitrary inquiries into the contents of a user's traffic. They note that some randomization might be added to enhance this defense. PGPA does involve some level of trust in the ISP. Furthermore, this approach binds packets to addresses (allocated by the ISP), but it might not guarantee that a particular individual intentionally sent some specified network traffic.

PGPA itself enables **detection** of violations. Combined with the ISP's records, it produces **evidence** that a policy-violating packet was sent across a link associated with a particular customer. Sufficient **identity** is required to establish an account with the ISP. We take this as a limited form of identity, and say that it is known only by the ISP. **Violations** are disclosed, and **violators** are identified as such, but only to the ISP and the parties to which it disclosed these things; we take that to be a limited. The system is **decentralized**, with and without violations, in that only the ISP associated with a particular link is involved in accusations about that link. There is no punishment as part of this system.

Packet passports, visas, and permits

Liu, Yang, Wetherall, and Anderson (Liu *et al.*, 2006) presented a system of "packet passports" to "authenticate the source of a packet at line speed[.]" This is secure in the sense that "that the [identity] of a source cannot be forged even if part of the routing system is compromised or has not deployed the authentication mechanism." Each pair of Autonomous Systems (ASes) that implement packet passports share a key. Packet passports use Message Authentication Codes (MACs) generated with

keys shared between the source AS and each of the ASes on the route to a packet's destination. These allow the intermediate ASes to verify the source AS for the packet. These passports are unforgeable, meaning that they are also usable as evidence in case of an attack. The recipient of unwanted traffic can present the traffic to the sender's AS and ask it to block further traffic. While Liu *et al.* do not focus on "accountability" explicitly, their passport system is motivated by related goals,[1] and it informs a number of later systems for tying network traffic to its source.

Although Liu *et al.* do not specify a punishment mechanism, a goal of packet passports is for the recipient of unwanted traffic to be able to provide **evidence** to the sender's AS to convince it to block additional traffic. This action on the sender's AS serves both as **punishment** and as **prevention** of further attack. The system envisions the establishment of shared keys via Diffie–Hellman exchanges through the routing infrastructure, making use of some authority to certify the public values; we take this as the **identity** requirement for participation. While the techniques for determining whether something is a violation and then punishing it are not specified in detail, they do not presume broad disclosure of the fact of the **violation** or the identity of the violator (although such broad disclosures are possible). The key authority might be centralized or decentralized (as in, *e.g.*, a web of trust); we take the passport system to be **decentralized**, regardless of whether there is a violation. Furthermore, Liu *et al.* note that it is incrementally deployable.

We also note two related mechanisms that are more focused on prevention than post-violation goals, although they use accountability-related language in describing their goals. As a result, we omit them from our summary of systems and mechanisms in Sec. 3.5. At a high level, both of these approaches provide some level of binding of traffic to its source.

Liu *et al.* point back to the visa system of Estrin, Mogul, and Tsudik (Estrin *et al.*, 1989) and, implicitly, the earlier visas of Estrin and Tsudik (1987). Both of those approaches identify their roots in ideas of Reed

[1] For example, Liu *et al.* argue that "the ability to identify the sources of an attack alone may deter future DoS attacks."

that were documented by Mracek (1983). Estrin, Mogul, and Tsudik identify accounting for the cost of traffic as one of the motivations for visas, connecting this early work to "accountability" in the resource sense of the goals identified by Clark (cf. Sec. 2.2.7). Estrin and Tsudik identify the importance of binding identity to actions from a (financial) liability perspective, which connects to the sense of "accountability" as blameworthiness (Estrin and Tsudik, 1987, pp. 175–176):

> [G]enerally when an organization, A, connects to an external organization, B, A must agree to assume responsibility for the actions of persons and machines within its organization boundaries (*e.g.*, to stand by purchase orders or other contracts written by its employees). In particular, A must vouch for the authenticity of internal entities that are able to export packets to B. If A is not confident as to the identity of an internal entity, then A should not allow it to use the gateway. Alternatively, A should not agree to [Inter-Organization Network] connections for which the liability exceeds the level of confidence that A has in its internal access control mechanism.

The visas of Estrin and Tsudik are granted by the networks that might be transited by network traffic. A visa is given to a source to include in all of the traffic in a particular session. Traffic without the visas can be rejected.

Around the same time as the work on packet passports, Dong, Choi, and Zhang (Dong *et al.*, 2006) described the Lightweight Internet Permit System (LIPS). LIPS is a lightweight approach in which a source requests a "permit"—a keyed hash of, *e.g.*, the requester's IP address—from the destination. In subsequent traffic, the source includes the permit (along with a permit for the return traffic). Dong *et al.* generalize this to larger networks divided into zones; some permits would allow traffic to move between a host and a zone boundary, while other permits would allow traffic to move between zones. They describe this mechanism as providing "traffic accountability" (or "traffic-origin accountability"). This is not precisely defined, but it implicitly relates to binding traffic to its source.

3.1.2 Learning about network loss and delay

Argyraki, Maniatis, Irzak, Ashish, and Shenker (Argyraki *et al.*, 2007) propose AudIt, a mechanism for learning about and localizing loss and delay in the Internet. This work takes accountability to mean "performance feedback [that helps] establish whether providers (and peers) are adequately performing their duty." AudIt enables various administrative domains (ADs) to provide information to a traffic source about where the source's outgoing traffic is dropped in the network and where that traffic experiences delays.

AudIt is a lightweight approach in that the feedback is not guaranteed to be accurate; inconsistent feedback indicates the need for further investigation outside of the scope of AudIt. Argyraki *et al.* argue that, if an administrative domain X provides correct feedback, then none of its peers can blame loss or delay on X without producing inconsistent feedback (Argyraki *et al.*, 2007, Lemma 4.1). The idea is that the source needs to further investigate inconsistent reports, potentially requesting signed feedback and then identifying either a lying administrative domain or a problematic inter-AD link. In the latter case, it is left to the ADs involved to resolve the problem. The traffic source may send the signed reports to the ADs involved; if one lied about the other, then its lie is revealed to the other AD.

Argyraki *et al.* considered it "impractical [at that time] to produce secure logs" of behavior, contrasting with many of the other systems and mechanisms that provide detection or evidence. Furthermore, because their focus was on network participants who have entered into business relationships, it was sufficient to provide evidence of wrongdoing to the affected participants without providing proof to the entire network.

Earlier work by Argyraki, Maniatis, Cheriton, and Shenker (Argyraki *et al.*, 2004) describes "packet obituaries" and takes a similar approach to providing feedback in that inconsistent feedback would lead to further investigation. As noted in Sec. 2.2.5, this work takes accountability to be "the property that enables activities on a system to be traced to specific entities, which may then be held responsible for their actions." The focus is on tracing the dropping of packets and providing this information to Autonomous Systems (ASes) along the packets' path

and to the source of the dropped packets. Information about a packet is sent backwards along a path, indicating the last AS that received the packet; this is either the AS of the destination, if the packet is received there, or an intermediate AS, if the packet is dropped before reaching the destination's AS. Packet obituaries are explicitly not intended to be sufficient to prove to a third party that a particular AS dropped a packet. Instead, as with AudIt, the expectation is that the sender will conduct further investigation once a problem is identified and localized to involve two adjacent ASes.

AudIt and packet obituaries provide **detection**, but they intentionally do not provide evidence for third parties. Both approaches are informed by the assumption that there are contracts, with associated requirements, between ADs/ASes; we take this to be an assumption of broadly known **identity** for ADs/ASes that participate in the system. In AudIt, problems generally are disclosed to the traffic source, while an AD that falsely blames a peer is identified to that peer, so we classify **violations** and **violators** as being identified to a unique recipient. Packet obituaries send feedback to every AS along the path, so **violations** are identified to a limited set of recipients. As in AudIt, a lying AS can be identified as a **violator** to its peer. Both mechanisms are **decentralized**, with and without violations, across ADs/ASes. However, each AD/AS does need to have some centralized information about operations across the AD/AS. While punishment does not figure prominently in these mechanisms, the contractual relationships between networks are part of the motivation for them. As a result, we view any related **punishment** as external to the mechanism and not requiring ongoing involvement (in the sense of carrying traffic) by policy violators.

3.2 Evidence to present to external parties

We now move to mechanisms whose focus is on producing evidence that can be used to convince third parties of something. Typically, this is evidence that a particular participant in a system did, or did not do, a particular action. This typically also provides detection of violations, but we distinguish these mechanisms from those in Sec. 3.1 because the ones reviewed here emphasize evidence for third parties. Some of

these mechanisms also provide some sense of judgment or blame, and even punishment, but those are also not the primary emphases of these mechanisms.

3.2.1 Accountable network storage

Yumerefendi and Chase (2007) described and implemented CATS, a certified accountable tamper-evident storage service. They provided definitions of *accountability* and *strong accountability* (see Def 2.1 above) capturing whether misbehavior can be determined with or without relying on claims by other (possibly compromised) participants. Yumerefendi and Chase also envision CATS as a building block for other services that provide their sense of accountability; these include transactions, access control, and service-based computation.

In the CATS system, participants read and write to a global storage system. The instructions for writing are signed and indicate what is being overwritten, thus chaining the writes together. The storage system can demonstrate that it has maintained and provided the correct state by producing the sequence of writes that lead to it. It can also prove that particular data were not part of the global state by providing a construction of the global digest that does not make use of the data in question.

CATS provides **detection** of misbehavior. It provides **evidence** that should be sufficient to convince a third party (*e.g.*, another user of the storage service or an auditor)—Yumerefendi and Chase note that "a participant's guilt is cryptographically provable to all participants."[2] CATS does prevent some improper actions, but those are more a function of its role as a storage service than of its accountability properties. Because clients cannot repudiate their actions or the effects of those actions on the system's shared state, the system also provides some measure of **blame** or **judgment**.

CATS assumes that participants have **identities** in the form of public keys. **Violations** and the identity of **violators** can be learned

[2]The expectation that the third party can verify this *independently* suggests that the third party may be more likely to present its own challenge to the server in order to obtain the evidence directly.

by any participant in the system. The storage functionality itself is described as implicitly centralized, but the accountability aspects of CATS can be **decentralized**, regardless of whether there is a violation. (A trusted auditor could be used, but is not required.) Finally, punishment is separate from this system.

3.2.2 PeerReview and its extensions

Haeberlen *et al.* (2007) describe and implement the PeerReview mechanism, which has significantly influenced subsequent work. They seek to collect evidence that allows participants to prove the commission of violations and to identify participants who have omitted required actions. As noted in Def. 2.4, they view an "accountable system" as one that "maintains a tamper-evident record that provides non-repudiable evidence of all nodes' actions." In the asynchronous setting considered by Haeberlen *et al.*, the possible violations are not responding to a message to which a response is prescribed by the protocol or sending a message that is not prescribed by the protocol. The potential for message delays means that the former cannot be conclusively proved; this gives rise to a distinction between suspicion and certainty, both of which are included in PeerReview.

Haeberlen *et al.* view applications of their notion of accountability as deterring faults, detecting faults, and assigning blame. Their sense of "accountability" *facilitates* the assigning of blame, in contrast with some other notions in which correctly assigning blame is all or part of what it means for a system to be "accountable." Haeberlen *et al.* (2007, pp. 178–179) identify two main goals of PeerReview (these are based on goals described by Chandra and Toueg (1996) for fault detectors):

> **Completeness:** (1) Eventually, every detectably ignorant node is suspected forever by every correct node, and (2) if a node i is detectably faulty with respect to a message m, then eventually, some faulty accomplice of i (with respect to m) is exposed or forever suspected by every correct node.

> **Accuracy:** (1) No correct node is forever suspected by a correct node, and (2) no correct node is ever exposed by a correct node.

PeerReview achieves these goals by using hash chains to log messages, and it introduces "authenticators" that prove to the sender and receiver of a message that the other party has received/sent the message and properly logged it. This evidence is shared in the network. Importantly, participant behavior is assumed to be deterministic, and other participants are assumed to have reference implementations of the protocol that any given participant is supposed to follow. This allows the shared evidence to be replayed to verify that the messages sent by a participant are correct. If a participant does not respond to messages, it is not provably faulty; it may just be down. In that case, other nodes are guaranteed to suspect it of having failed until it resumes responding to messages. Other natural limitations are that only faults that affect messages are detectable and that they must be detectable by a correct node (*e.g.*, two colluding nodes might exchange bad messages between themselves but appear correct to all other nodes).

PeerReview provides **detection**, **evidence**, and **judgment**, the last through the PeerReview modules' suspicion or certainty that a node is violating the protocol. The mechanism assumes public keys that are bound to a unique **identity**, but Haeberlen *et al.* note that "any type of name binding that avoids Sybil attacks" suffices. These identities are known throughout the network. **Violations** are disclosed, and **violators** are identified as such, both broadly. Regardless of whether there is a violation, PeerReview is **decentralized**. There is no punishment involved. PeerReview requires digital signatures.

Haeberlen, Avramopoulos, Rexford, and Druschel (Haeberlen *et al.*, 2009) build on PeerReview to describe NetReview, a mechanism for detecting problems with the Border Gateway Protocol (BGP). NetReview guarantees that observable faults (the faults that can reasonably be detected by non-faulty ASes) are "eventually detected and irrefutably linked to a faulty AS and that [...] no verifiable evidence is ever generated against a non-faulty AS." From a high-level accountability perspective, this is very similar to PeerReview, using tamper-evident

logs inspired by the latter. However, BGP-specific considerations require many extensions in the implementation that we do not consider here, and we do not include NetReview in our summary tables in Sec. 3.5.

Backes, Druschel, Haeberlen, and Unruh (Backes *et al.*, 2009) consider how to provide randomness for "accountable" protocols in the sense of PeerReview. They describe the CSAR protocol to provide cryptographically strong, accountable randomness, and they implement this as a library that can be used by PeerReview. CSAR provides a way to audit the generation of random values that might be used in a protocol while still providing strong randomness. In particular, Backes *et al.* (2009, p. 2) state that it

> satisfies the following requirements:
>
> 1. The pseudo-random generator should output cryptographically strong randomness. It is not sufficient for the output of the generator to be uniformly distributed. We require that the node generating the output should only be able to compute values it could also compute if the output was truly random.
> 2. The pseudo-random generator should be accountable, *i.e.*, after each random value r is generated, it should be possible to generate a proof that this value r was indeed correctly derived from a given seed. Thus, if a node generates a value incorrectly, it can be held accountable because it cannot produce a valid proof.
> 3. Future random values of correct nodes should be unpredictable, *i.e.*, to a node that learns random values $r_1, \ldots, r_i$ and the corresponding proofs, all future random values $r_{i+1}, \ldots$ should still look random. This excludes the obvious solution of using the random seed as a proof.
> 4. Properties 1–3 should hold even if malicious nodes are present while the seed is computed. In particular, no node should be able to influence the output of its own generator by choosing a suitable seed.

CSAR maintains these properties by combining the application of a trapdoor one-way permutation and hashing. Values s_i are produced in sequence using the inverse of the permutation, which can be computed knowing a secret key. This allows for public verification that s_{i+1} was derived from s_i once both are known. The random value r_i is derived from s_i using the hash function, which serves to break the functional relationship between the s_i values. Both s_i and r_i are written to the audit log.

The approach that Backes *et al.* take means that they can use PeerReview, instead of "a separate mechanism to detect if a node breaks [CSAR's] randomness[-]generation protocol or ignores a coin-toss message." We view CSAR as an extension to PeerReview (or other mechanisms) rather than a separate accountability mechanism, so we do not include it in our summary tables in Sec. 3.5.

Haeberlen, Aditya, Rodrigues, and Druschel (Haeberlen *et al.*, 2010) propose Accountable Virtual Machines (AVMs) based on the tamper-evident logs of PeerReview. By contrast with PeerReview, AVMs allow software to run without modifications on a VM; like PeerReview, a distributed system can be audited by auditing its component nodes. In AVM scenarios, a customer would like to have a provider run software for her. The provider's system (in particular, an Accountable Virtual Machine Monitor) maintains a tamper-evident log of all incoming and outgoing messages as well as any nondeterministic events that might affect the customer's software. The customer audits the provider's work by replaying the log (assuming it passes integrity checks) on a reference implementation of the VM and software that she maintains. She determines whether or not there is a correct execution of the reference implementation that matches the log. In particular, correct behavior has an existenial aspect to it—the question is whether the reference implementation "*can* produce the same network output as [the provider's machine] when it is started in the same initial state and given precisely the same network inputs" (emphasis added).

Haeberlen *et al.* identify detection and evidence as explicit goals of AVMs; the log and reference implementations provide evidence that can be given to a third party if the customer's checks fail. AVMs provide completeness and accuracy properties. These update the analogous properties of PeerReview to the AVM setting.

Haeberlen *et al.* implemented AVMs to detect cheating in multiplayer games and analyzed AVM performance in that context. They also suggest applications in distributed systems generally, malware detection via traffic analysis, and execution of programs in the cloud.

AVMs are very similar PeerReview in our classification in Sec. 3.5. Haeberlen *et al.* present AVMs in a two-party case, in which one party would audit software running on the other party. However, when more parties are involved, information about the software's communication may be needed from many parties, and the information about the violation/violator may be distributed broadly.

Ó Coileáin and O'Mahony (Ó Coileáin and O'Mahony, 2014a; Ó Coileáin and O'Mahony, 2014b) present Svant, an architecture for providing accountability in content-distribution networks. The architecture adds accountability agents throughout the network to collect information (*e.g.*, performance metrics, experience metrics, and demographics) about content delivery and accurately report it back to the content provider. It makes use of cryptographic tools such as hashes and digital signatures, which they note is similar to the approach previously taken by PeerReview.

Ó Coileáin and O'Mahony (2015) also survey approaches to providing non-repudiable evidence about content distribution in various content-distribution architectures. They highlight PeerReview as an important accountability solution for peer-to-peer (P2P) systems. They identify other systems as having similarities to PeerReview but working in other content-distribution models: Reliable Client Accounting (RCA) of Aditya, Zhao, Lin, Haeberlen, Druschel, Maggs, and Wishon (Aditya *et al.*, 2012) for hybrid CDN–P2P systems, and their own Savant architecture for use with information-centric networking. Ó Coileáin and O'Mahony also identify Repeat and Compare (see Notes at the end of this chapter) as an accountability mechanism for content-distribution networks.

3.2.3 Cryptographic commitments and zero knowledge

A couple of lines of recent work have studied the use of cryptographic commitments and zero-knowledge arguments or proofs to provide ac-

countability for actions or processes that are not fully transparent. The general idea is that the actors who will be "held accountable" publish cryptographic commitments binding themselves to certain actions and policies. They then demonstrate in zero knowledge that these commitments show proper behavior without revealing the details of that behavior.

Kroll *et al.* (2017) have recently studied accountable algorithms from both computer-science and legal perspectives. Kroll *et al.* leave "accountability" undefined in a formal sense. They argue against "accountability" as transparency, saying that the latter "may be undesirable, such as when it discloses private information or permits tax cheats or terrorists to game the systems determining audits or security screening."

Instead, Kroll *et al.* (2017, p. 656) argue for

> *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.

Towards this end, they propose using cryptographic commitments and zero-knowledge proofs. A decision maker would commit to a policy in advance of making a decision and then commit to the inputs and outputs of the procedure once the decision was made. The zero-knowledge proofs would be used to ensure that the committed-to procedures and values were actually used. The decision maker might withhold the details of the decision-making procedure; even so, interested parties—like those affected by the decisions—could ensure that the same procedure was used for each decision.

This approach is described in very broad terms by Kroll *et al.*, although they do discuss aspects of how it might be implemented. A mechanism following this approach would provide **detection** and **evidence**. The decision maker would likely need to be identified, at least enough to make the cryptographic commitments and participate in the zero-knowledge proofs. **Violations** would be observed, and the decision-maker would be identified as a **violator**; the extent to which this information would be known would depend on how the proofs were provided. (Kroll *et al.* (2017, p. 672) say that "[t]hese zero-knowledge proofs could either be made public or provided to the system's decision

subjects along with their results.") Because of the broad description of this approach, we omit most of the dimensions from our classification of it in the summary tables in Sec. 3.5.

Frankle, Park, Shaar, Goldwasser, and Weitzner (Frankle *et al.*, 2018) design, implement, and evaluate a mechanism to provide accountability for, *e.g.*, non-public court orders. (This is inspired by a paper-based proposal made by a U.S. magistrate judge.) They view accountability as involving detection (in the sense of Sec. 2.1.1) and responsibility (determining whether rules were followed). They aim to do this by ensuring that (Frankle *et al.*, 2018, Sec. 3.2)

> enough information [is revealed] to the public that members of the public are able to verify that all surveillance is conducted properly according to publicly known rules, and specifically, that law enforcement agencies and companies (which we model as malicious) do not deviate from their expected roles in the surveillance process. The public must also have enough information to prompt courts to unseal records at the appropriate times.

Their approach is to have different parties in the court-order process (courts, law-enforcement, and corporations in receipt of orders) post cryptographic commitments and, when applicable, zero-knowledge arguments to a public, append-only ledger. This allows the public to verify the propriety of actions taken. Frankle *et al.* also include a secure multiparty computation component in their mechanism. This allows the public to learn aggregate statistics about the non-public actions.

Aspects of the work of Frankle *et al.* resemble earlier work of Goldwasser and Park (2017), which used cryptographic commitments to a public blockchain (they note that Ethereum is sufficient in its current form) and zero-knowledge arguments to audit the contents of those commitments. This is motivated by ensuring compliance with non-public regulations, with courts resolving disputes between the organizations that publish commitments and the auditors who check them. As noted by Goldwasser and Park, the commitment aspect of this resembles timestamping services.

3.2.4 Timestamping services and extensions

Buldas, Laud, Lipmaa, and Villemson (Buldas *et al.*, 1998) described timestamping schemes with accountability-related properties. One such property is *relative temporal authentication*, which is the ability to verify a claim that one of two timestamped objects was stamped first, without having to trust the timestamping service. Another property is *detection of forgeries*, which they define as "(1) [determining] whether the timestamps possessed by an individual have been tampered with and (2) in the case of tampering, [determining] whether the timestamps were tampered [with] by the [Time-Stamping Service] or [later.]" If the tampering was done after the timestamping protocol's execution terminated, then the forgery-detection mechanism is unable to assign blame. One of their motivating goals is "to make users liable only for their own mistakes[,]" which fits in with some notion of accountability. The approaches of Buldas *et al.* strengthened the properties of earlier timestamping systems and satisfied efficiency goals that they argued could not be substantially improved. This work informed more recent efforts that were more directly concerned with accountability.

In subsequent work, Buldas, Lipmaa, and Schoenmakers (Buldas *et al.*, 2000b) explicitly describe an "accountable" timestamping service. They say that accountability has been achieved "if the next two properties hold whenever there is at least one uncorrupted party (say, one honest client interested in legal value of a particular time stamp) during the creation of stamped information:

- Fraud detection: The service makes the trusted third parties accountable for their actions by enabling a principal to detect and later prove to a judge any frauds affecting the relative ordering between timestamps.
- Anti-framing: If a party has honestly followed the protocol but is still accused in forgeries, she can explicitly disavow any false accusations.

This work improves the protocols of Buldas, Laud, Lipmaa, and Villemson and adds an accountable publication protocol that removes the need to audit the publication process. Auditing is enabled by the part

of a timestamping service in which certain signatures are published in a world-readable way, *e.g.*, in a newspaper.

The work of Buldas, Lipmaa, and Schoenmakers provides **detection** and **evidence**. There is a **centralized** timestamping authority as well as the authenticated publishing medium. The **identities** of the trusted parties, and potentially of the clients, are broadly known via their public keys. **Violations** are disclosed, and **violators** are identified as such, to a limited set of parties; the principal obtains evidence that she can take to a third party. There is no punishment in the mechanism.

Other work by Buldas, Laud, and Lipmaa (Buldas *et al.*, 2000a) builds on the work of Buldas, Lipmaa, and Schoenmakers to manage key certificates in a way that achieves the same accountability goals as timestamping services. In particular, they require that "every validity change of a certificate [be] accompanied by a transferable attestation" of this change. As in the timestamping work they build upon, forgeries are detectable, and false accusations are refutable. In constructing their mechanism, they propose an "undeniable attester" primitive whose existence is equivalent to the existence of collision-resistant hash functions.

3.3 Judgment or blame

We now review approaches to blaming system participants for violations or rendering judgment that a participant has committed a violation. This includes high-level approaches to auditing and accountability-related properties for anonymous groups.

3.3.1 Accountability through audit

We use "audit" in a broad, informal sense that is generally consistent with the definitions from the literature noted in Chap. 2. In particular, we view audit mechanisms as involving some sort of review of records in order to assess what has happened in a system. This might be done by some combination of humans and computers. The process might involve either an unstructured collection of data—*e.g.*, all records pertaining to the system, whether on paper or disk—or a very structured collection

of records—*e.g.*, signed, timestamped records of messages sent and received, written to a trustworthy log file.

We review a couple of audit-focused approaches here. These are higher level than many of the mechanisms discussed above, but we can still give rough classifications of them. They differ from each other in whether the auditors are internal or external participants in the mechanism. As discussed below, particular implementations of these might, *e.g.*, disclose identity more broadly than is required by the high-level architecture. Other approaches also contain elements of auditing even though we categorize them differently. For example, as discussed in Sec. 3.2.2, PeerReview securely logs actions and has nodes across the network review those actions. We view PeerReview's focus as being on the provision of evidence, while we view the focus of the architectures we review in this section as instead being on how auditors assess evidence.

As one exemplar of accountability through audit, we consider the work of Jagadeesan, Jeffrey, Pitcher, and Riely (Jagadeesan *et al.*, 2009), who describe a formal operational model for distributed systems that achieve accountability in this way. The auditor(s) may "blame" a set of participants for a violation, *i.e.*, name the members of that set as potential violators. This gives rise to multiple questions about the properties of the audit protocol (such as whether everyone blamed is a violator and whether all non-violators are able to ensure that they are not blamed). These properties are treated as accountability properties, but they do not change the underlying approach of blaming (sets of) individuals for violations.

In the model of Jagadeesan *et al.*, (sets of) individuals are **blamed** for violations. The auditors rely upon evidence to make their judgments, but the notion of accountability captured by this framework fits better with blame or judgment. Because sets of individuals are blamed using their identities, some sort of **identity** is required to participate. These identities might not be broadcast throughout the system, but they are used for communication between participants, so we expect these to be broadly disclosed. The existence of a **violation** is known by the auditor and potentially by selected other participants, *e.g.*, those who have forwarded a message that constitutes a violation. **Violators** are identified as such by the auditor, but they are not necessarily identified

more broadly. While auditors are trusted, they do not have a global view (*i.e.*, they interact with the system as participants), so we view this approach as **decentralized**, both with and without violations. There is no punishment involved in this system. The auditors can render their judgment without the ongoing **participation** of the violator.

As a second exemplar of accountability through audit, we note the work of Barth, Datta, Mitchell, and Sundaram (Barth *et al.*, 2007), who defined a logic for utility and privacy that they applied to models of business practices. (We discuss this logic in Sec. 4.1.3.) In their application to healthcare practices, agents in the system are responsible for tagging messages (*e.g.*, to ensure that sensitive health information is not forwarded to the agents responsible for scheduling patient appointments). Different roles in the system have associated responsibilities. An agent playing a particular role must then fulfill those responsibilities. Barth *et al.* say that an agent is accountable for a policy violation if the agent performed an action that occurred before the violation (from some perspective on the system's behavior) and also did not fulfill his responsibilities. They then give an algorithm to identify accountable agents (via communication logs). While an "accountable agent" might not be the cause of the violation in question, causality or the lack thereof can be determined by a human auditor. The auditor can check the correctness of message tags and can repeat the process until the agent who caused the violation is identified.

Barth *et al.* do not describe a specific implementation of the mechanism they design, but they do analyze a healthcare provider's patient portal using their approach. Their approach provides **detection** and **blame** for violations. If the audit logs are secure, they might be usable as **evidence** of policy violations. However, providing evidence to third parties (beyond the auditor) is not a goal of this work. Participants' **identities** must be at least known to the auditor. In many cases, these identities are known more broadly as well, *e.g.*, when a doctor receives a question from her patient, both participants know with whom they are communicating. We thus expect these to be broadly known. **Violations** are disclosed to the auditor, who also knows who the **violators** are. Here, the auditing engine, which is used even in the absence of a violation, and the human auditors, who determine whether an agent is the

cause of a violation, appear to be **centralized**. There is no punishing entity.

3.3.2 Blame in anonymous groups

Blame in anonymous communication

DISSENT (DIning-cryptographers Shuffled-Send NeTwork) is an anonymous-communication protocol for well defined groups of participants whose membership is closed and known to its members. This was introduced by Corrigan-Gibbs and Ford (2010) and then modified by Syta, Corrigan-Gibbs, Weng, Wolinsky, Ford, and Johnson (Syta *et al.*, 2014) to fix various flaws. DISSENT supports a notion of "accountability" that ensures "that any disruption results in the identification of some malicious member during a 'blame' process."

DISSENT enables members of such a group to send anonymous messages—to each other in a point-to-point fashion, to the whole group in a broadcast fashion, or to someone who is outside of the group—in a manner that enables the receiver to determine that *some* group member sent the message but prevents the receiver from determining *which* member sent it. It is suitable for medium-sized groups, but Wolinsky, Corrigan-Gibbs, Ford, and Johnson (Wolinsky *et al.*, 2012) have separately studied how to scale this to thousands of users. Application scenarios in which this form of anonymous communication is useful include twelve-step programs, "board-room scale" votes, online discussion groups that focus on sensitive topics (sexuality, politics, religion, disease, *etc.*), and whistle blowing.

After the encrypted messages are sent, DISSENT enables senders to determine whether each of the sent messages was among the messages received. If so, the received messages are decrypted, and the protocol terminates. This builds on the verifiable, anonymous shuffle protocol of Brickell and Shmatikov (2006). If at least one sender sent an encrypted message that was not received uncorrupted, all protocol participants are called upon to broadcast previously private information needed to prove that they behaved correctly in previous protocol phases. At least one member we be unable to do so and will thus be exposed as malicious. More specifically, it is "active" malicious behavior that the

blame phase of DISSENT will expose, *e.g.*, corruption or blocking of other participants' messages, denial-of-service or "spamming" attacks on the group, or the creation of an arbitrary number of Sybil identities or sock puppets that subvert the group's deliberations. In a DISSENT execution that terminates with the exposure of a malicious participant, the honest participants destroy keys that are needed to decrypt the sent messages; thus, no plaintext messages are disclosed to the malicious participant.

DISSENT provides **detection**, **evidence**, and **judgment** for dishonest behavior. Participants are required to have **identities** in the form of public keys, and they are assumed to be unable to easily leave and rejoin the system or otherwise create Sybils. **Violations** are disclosed, and **violators** are identified as such, both broadly. DISSENT is **decentralized**, both with and without violations. There is no punishment mechanism as part of DISSENT.

Anonymous and accountable communities

Farkas, Ziegler, Meretei, and Lörincz (Farkas *et al.*, 2002) describe an approach to "anonymous accountability." Their Anonymous and Accountable Self-Organizing Communities (A2SOCs) involve principals who may take on multiple identities within the system. The A2SOC enables interaction among the participants as well as accountability properties. Farkas *et al.* use "internal accountability" to mean that the in-system identity "is identifiable within the group and can be held responsible for his/her actions according to the [...] policy of the group." They use "external accountability" to mean that the principal "behind the [in-system identity] is identifiable and can be held responsible for his/her actions according to the [...] law of the external environment hosting the group." More generally, A2SOCs achieve the following goals (Farkas *et al.*, 2002, p. 85):

1. Participants of an A2SOC want to interact with each other effectively [...]
2. [P]articipants of an A2SOC may want to remain anonymous. That is, each [principal] needs a virtual identity

> and the backward mapping from the virtual [identity] to the [principal's] identity has to be protected.
>
> 3. Participants of an A2SOC may want to be unambiguously identifiable via their virtual identity within the collaboration group for purposes of gaining reputation, paying credits, *etc.* [...] For these purposes accountability needs to be optionally maintained: billing requirements must be enforceable, misbehaviors must be punished, *etc.*
> 4. Fraud must be prevented.

Farkas *et al.* provide protocols to implement these notions of accountability. These rely on a trusted-computing base (TCB) and threshold cryptography. This ensures both that the TCB must participate in any disclosure of the mapping between a principal and her in-system identities and that the community of users must also be involved in such a disclosure. These protocols ensure that, "without revealing the identity of the [principal,] it can be determined whether two [in-system identities] belong to the same [principal]." They also allow revelation of the principal's identity under appropriate conditions.

A2SOCs provide **evidence** and **blame** in the form of identifying violators. In both the internal and external senses of "accountability" used by Farkas *et al.*, this enables mediated **punishment**. The sample protocols include the possibility of the community terminating identities associated with the principal, who remains unidentified, as a penalty; this provides punishment in the internal-accountability scenario. Participation requires that **identity** be disclosed to the TCB. **Violations** are disclosed, and **violators** are identified as such, to the broad community of participants in the internal-accountability scenario. In the external-accountability scenario, the sample protocols only identify the violator to "the authorities." This might be viewed as a limited set, but a conviction in a public court would disclose the violator's identity broadly. The TCB means that the system is **centralized**, both with and without violations. The **punishing entity** is either internal or external, depending on the sense of accountability being considered. In the case of internal accountability, **ongoing involvement** is required.

Farkas *et al.* point to earlier work, including that of Buttyán and Hubaux (1999), for related approaches. Buttyán and Hubaux discussed a high-level model of anonymous tickets for services. Anonymous tickets are ones in which the customer is not bound to the ticket, but the service provider might or might not be. Buttyán and Hubaux identify four classes of tickets, depending on which of the customer and service provider are bound to a ticket. (They give cash as an example of a ticket that binds neither the customer nor the provider.) They do not provide a precise definition of accountability, but it is implicitly bound up with identification of users. In particular, Buttyán and Hubaux say that "[i]f necessary, anonymity can be revoked and users can be held accountable for their malicious behaviour." We do not further classify mechanisms that might implement this architecture.

3.4 Punishment

We now turn to systems and mechanisms that we view as focused on punishment. Approaches to punishment include reducing either anonymity or reputation as well as preventing a violator from participating in the system in the future. We also review a number of approaches that incentivize correct behavior.

3.4.1 Diminished anonymity

E-cash was pioneered by David Chaum in the early 1980's (Chaum, 1982; Chaum, 1983). It was designed to have the anonymity and untraceability properties of physical cash: A user should be able to withdraw money from the bank and spend it with a merchant without revealing her identity, and the merchant should be able to deposit the money received into his account without the bank learning how individual users spent their money. Because of the replicable nature of the strings of bits that represent digital money, a central challenge in realizing e-cash is how to prevent or deter "double spending," in which users or merchants make and spend (or deposit) multiple copies of electronic coins.

Chaum's solutions to this challenge, and many others that grew out of his work, assume that the bank is a centralized party that checks

for double spending. Chaum's initial proposals (Chaum, 1982; Chaum, 1983) were "on-line," in the sense that the bank must be involved in every transaction in order to prevent double spending. Chaum, Fiat, and Naor (Chaum *et al.*, 1990) introduced "off-line" e-cash, in which double spending was not strictly prevented, but the identity of double spenders would be revealed by the bank after-the-fact, along with incontestable proof of the violation. This off-line mechanism also protects against a cheating merchant who might try to collude with a customer in order to allow undetectable double spending or attempt to frame an innocent customer as a double spender.

While a complete survey of e-cash schemes is beyond the scope of this paper, we note that there have been many proposals that take different approaches and provide different properties, including differences in prevention *vs.* detection, centralization *vs.* decentralization, and security *vs.* efficiency. An interesting example in trading off security and efficiency is Rivest and Shamir's MicroMint (Rivest and Shamir, 1997), which is designed so that small-scale fraud will be unprofitable, while large-scale fraud will be detectable.

An exemplar of the off-line approach, proposed by Camenisch, Hohenberger, and Lysyanskaya (Camenisch *et al.*, 2006), explicitly addresses accountability as a goal to be balanced with privacy, while extending the accountability goals beyond double spending. Specifically, in addition to detection of double spending, their work supports spending limits for each merchant, motivated by concerns that anonymous e-cash can allow undetectable money laundering. A user's anonymity and untraceability is guaranteed as long as she does not violate either policy (double spending or spending limits). Violations can be detected, and it can be determined whether a user or a merchant cheated. When a violation is detected, the bank becomes (mathematically) able to identify the violating user as well as trace the other activities of the violating user. Camenisch *et al.* frame this as punishment mediated by the bank.

The system of Camenisch *et al.* requires **identity** in the form of a public key. It provides **detection** of violations, **blame** for violations (by identifying the violators), and **evidence** of violations (in that anybody can run a protocol to verify the violations). It also provides some measure of (mediated) **punishment** in the form of lost anonymity. **Violations**

are disclosed, and **violators** are identified as such, broadly. This system is **centralized**, by virtue of the bank, both with and without violations. The **punishing entity** is the bank, who identifies the violator, but any system participant can run a protocol to verify the bank's identification. This is internal to the system. It does not require the violator's ongoing **participation** in the system.

3.4.2 Diminished reputation

Diminished reputation as a result of a policy violation might be viewed, in itself, as a punishment for the violation. However, the main punishing effect comes through its inhibition of future interactions. This effect could include other system participants declining to interact or agreeing to interact only under conditions that are less favorable to the party with diminished reputation. However, if a party expects they will never again have an interaction that is covered by a reputation system, the punishment and deterrence due to a diminished reputation go away. The punishing effects of diminished reputation, and the associated deterrence, connect reputation systems to accountability, even when the designers of a reputation system or mechanism do not explicitly motivate it by "accountability."

Even a partial survey of reputation systems is beyond the scope of this survey. Here, we note some general principles and consider an exemplar, but we defer to the surveys noted below for more thorough reviews of this area.

Resnick, Zeckhauser, Friedman, and Kuwabara (Resnick *et al.*, 2000) provide an early review of reputation systems that highlights some of the key properties of such systems. In general (Resnick *et al.*, 2000, p. 46),

> when people interact with one another over time, the history of past interactions informs them about their abilities and dispositions. Second, the expectation of reciprocity or retaliation in future interactions creates an incentive for good behavior. (Political scientist Robert Axelrod calls this the "shadow of the future".) An expectation that people will

> consider one another's pasts in future interactions constrains behavior in the present.

This last point captures the deterrent effect; the question is how to produce this in interactions among parties who do not know each other. They succinctly identify this as the goal of reputation systems as follows (Resnick *et al.*, 2000, p. 46):

> Reputation systems seek to establish the shadow of the future to each transaction by creating an expectation that other people will look back on it.

One recent survey in this area is by Granatyr, Botelho, Lessing, Scalabrin, Barthès, and Enembreck (Granatyr *et al.*, 2015) on trust and reputation models. Another recent survey, by Hendrikx, Bubendorfer, and Chard (Hendrikx *et al.*, 2015), focuses on reputation systems themelves; it provides a review and taxonomy of such systems. Jøsang, Ismail, and Boyd (Jøsang *et al.*, 2007) gave an earlier survey of trust and reputation systems; that included a more extensive discussion of the nature of trust involved.

As an exemplar of a reputation mechanism, we consider the one for mobile ad-hoc networks proposed by Buchegger and Le Boudec (2003). Each node i in a network has, for each other node j that it tracks, a trust rating and a reputation rating. The reputation rating, which affects how i behaves towards j, is affected by both i's direct interactions with j and information obtained about j from other nodes (in particular, nodes that i trusts or that have experiences with j that are similar to i's experiences). If i's view of j is sufficiently bad, then i will avoid routing through j, and i will ignore future route requests from j. We view this as punishing j for misbehaving in the routing protocol, and Buchegger and Boudec explicitly note that they do *not* punish nodes that give inaccurate reports to the reputation mechanism. The particular (modified Bayesian) approach to updating reputation is unrelated to the accountability properties of this system.

The mechanism of Buchegger and Le Boudec provides **punishment** through the avoidance of a node in routing and ignorance of its route requests. Arguably, it also provides a sort of **judgment**, in an average

sense, over many different violations and non-violations. The mechanism requires that violations are detected, but it propagates that information instead of actually doing the detecting. This mechanism requires **identities** that are known broadly; Buchegger and Le Boudec note that these must be "persistent, unique, and distinct." In many ways, the *point* of a reputation system is to identify **violators** in a fairly broad way and to disclose the **violations** broadly. This mechanism is **decentralized**, both with and without violations. The punishment is meted out through the other nodes in the network, making these internal **punishing entities**. Because punishment takes the forms of routing around and ignoring the violator, it relies on the continuing **participation** of the violator.

3.4.3 Banishment from the system

We now consider mechanisms that effect punishment through banishing a participant from a system, either by revocation of a credential needed for participation or by having other participants refuse to interact with a known violator. In such cases, the violator might be able to return to the system under a different identity, but he could no longer use the identity that he used to commit the original violation.

Like e-cash systems, applications and protocols that use anonymous blacklisting mechanisms allow anonymous participation. In contrast to e-cash, participants in these systems are not identified when they commit a violation; instead, they are blacklisted (*i.e.*, their credentials for participation are revoked) without being identified. Henry and Goldberg (2011) have recently surveyed this space of mechanisms and identified three broad subspaces thereof: pseudonyms, mechanisms like Nymble (Tsang *et al.*, 2011), and revocable anonymous credentials. Anonymous blacklisting provides varying levels of privacy (ranging from pseudonyms to complete anonymity without trusted third parties); however, as the privacy guarantees are strengthened, the feasibility of implementation decreases.

As an exemplar of these systems, we take the PEREA revocable anonymous-credential mechanism of Au, Tsang, and Kapadia (Au *et al.*, 2011), which extends preliminary work of Tsang, Au, Kapadia, and

Smith (Tsang *et al.*, 2008). A user registers with a service provider and obtains credential information from the provider. The user generates anonymous tickets and presents these to the service provider. She also uses zero-knowledge proofs to demonstrate that none of her recent tickets have been blacklisted by the provider; this ensures that the provider cannot link different tickets from the same user. The provider uses a cryptographic accumulator to store the blacklist of tickets corresponding to policy violations; a ticket must be blacklisted before a specified number of additional tickets from that user are accepted. PEREA avoids the use of a trusted third party and allows a user to ensure that she has not been blacklisted before sending a request to a service provider. Au *et al.* also describe a variation in which different violations may be given different weights; a user is then allowed to authenticate to a service as long as the sum of her recent violations is below a specified threshold.

PEREA is a **punishment** mechanism that provides mediated punishment. The service provider must detect violations and judge that the user should be punished through other means. The user is required to register a public key **identity** with the service provider; individual service providers might require that this be bound to an offline identity, but that is not inherent in the mechanism. **Violations** are known to the service provider (and the user), but not to any other party. Importantly, **violators** are not identified as such. PEREA is **decentralized**, with and without a violation, although the initial authentication of the user to the service provider might require external infrastructure in some applications. The **punishing entities** are part of the system. Punishment does require **ongoing** involvement by the violator in order to be meaningful.

3.4.4 Incentivizing correct behavior

We now review mechanisms that incentivize correct behavior in multiparty computation, lotteries, and other settings. First, we note some that assume the context of legal and banking systems to which digitally signed documents can be presented. Second, we review a number of recent mechanisms that use cryptocurrency frameworks—typically Bitcoin

(Nakamoto, 2008)—instead of courts and banks. These cryptocurrency-based mechanisms take the general approach of having a participant commit something of value that she can recover only after a specified period of time. If, in the interim, she violates a policy being enforced by the mechanism, another party is able to claim her deposit. In some ways, this seems close to automatic punishment; the policy violation makes it mathematically possible for another party to claim the violator's deposit, and that party will be incentivized to do so if the deposit is large enough. However, these mechanisms envision the claiming party taking some affirmative step to claim the deposit and doing so after learning of the violation, so we will still view these as providing mediated punishment. There may be ways that this active step could be avoided, although likely at some cost to the party who would claim the deposit in case of a violation. While many of these mechanisms are not framed explicitly in terms of "accountability," the near-automatic nature of the punishment that they provide, and the pseudonymous context in which they do so, makes them of interest to us.

Lindell (2008) presents a mechanism, which he notes is inefficient, to incentivize fairness in the context of a legal system. His mechanism has the property that (Lindell, 2008, p. 123)

> either both parties receive output (and so fairness is preserved) or one party receives output while the other receives a digitally-signed cheque from the other party that it can take to a court of law or a bank. This cheque can contain any sum of money, as agreed by the parties. The protocol further has the property that the only way that a party can evade paying the sum in the cheque is to reveal the other party's output, thereby restoring fairness.

Lindell points back to even earlier work by Chen, Kudla, and Paterson (Chen *et al.*, 2004), which informed the construction of his mechanism, for two things. One is the observations that, "in order for a signature to be *enforced*, it needs to be presented at a court of law" (Lindell, 2008). The other is the presentation of a system for "concurrent signatures" in the random-oracle model. This allows (Chen *et al.*, 2004, p. 287)

> two entities to produce two signatures in such a way that, from the point of view of any third party, both signatures are ambiguous with respect to the identity of the signing party until an extra piece of information (the keystone) is released by one of the parties. Upon release of the keystone, both signatures become binding to their true signers concurrently.

Lindell's mechanism provides a digitally signed check that can be presented to a bank for payment. As with the cryptocurrency-based mechanisms discussed below, this is an affirmative step that would qualify this as mediated punishment. Arguably, the process of presenting a check to a bank for payent requires greater involvement by the payee than does submitting a cryptocurrency transaction.

Herlihy and Moir (2016) propose approaches to distributed ledgers that would increase some accountability-related properties. They focus on permissioned ledgers, in contrast to permissionless ledgers like Bitcoin, and argue for reducing non-deterministic protocol choices. In this context, they propose cryptographic techniques to provide a one-way communication channel in which the sender gets confirmation of receipt and processing from the receiver of the messages, and the receiver gets confirmation that its confirmations were received (to avoid false accusations by the sender). They situate their work in the context of a proof-of-stake protocol in which a participant "posts a bond, in the form of tokens, which it will lose if it is caught violating the protocol. If [the participant] is caught, proof of that transgression is posted to the blockchain, the culprit is expelled and its bond confiscated." This work focuses on detecting violations, collecting evidence about possible violations, and providing evidence to external parties; the token-forfeit aspect provides punishment for violations. Herlihy and Moir (2016, p. 12) note that they "have described mechanisms for proving misbehavior in some cases and exposing evidence that may be accumulated and interpreted externally in others, while not addressing how this should be done." They contrast this with PeerReview, in particular its detection guarantee and its prescription of how participants should interact.

Andrychowicz, Dziembowski, Malinowski, and Mazurek (Andrychowicz *et al.*, 2013) present an approach to incentivizing correct behavior in

two-party protocols. This requires making Bitcoin transactions "nonmalleable." It builds on other work (Andrychowicz *et al.*, 2014; Andrychowicz *et al.*, 2016) on secure multiparty computation in which they use transactions that are "time locked"—*i.e.*, invalid before a specified time—to construct a secure lottery protocol. Bentov and Kumaresan (Bentov and Kumaresan, 2014a; Kumaresan and Bentov, 2014) define an ideal functionality ("claim-or-refund") that they implement in Bitcoin (pointing also to earlier work); they then use this to construct systems that punish violators in secure lotteries and general secure computation.

Kiayias, Zhou, and Zikas (Kiayias *et al.*, 2016) develop a protocol for multiparty computation that is incentivized to be both fair (if any party learns the output, then all do) and robust (the protocol guarantees delivery of the output, which in this setting also ensures that a denial-of-service attack imposes a monetary cost on the attacker). The robustness property strengthens the protocols of Andrychowicz *et al.* and of Bentov and Kumaresan. Like those protocols, the one of Kiayias *et al.* uses time-locked transactions; they describe an implementation in Ethereum. They also note that the fair protocols of Andrychowicz *et al.* and of Bentov and Kumaresan can be made robust by incorporating identifiable abort (pointing to the work of Ishai *et al.* (2014) that we discuss in Sec. 2.2.3).

Kosba, Miller, Shi, Wen, and Papamanthou (Kosba *et al.*, 2016) describe Hawk, "a framework for building privacy-preserving smart contracts" in blockchain systems. In addition to providing privacy guarantees, Hawk provides notions of "contractual security" that can include fairness in the sense considered by the protocols described above.

Ruffing, Kate, and Schröder (Ruffing *et al.*, 2015) use smart contracts in Bitcoin to address "equivocation," or making contradicting statements. They define an "accountable assertion" primitive that cryptographically binds a principal's statement to a context. A principal is "held accountable" in the following sense. She makes a time-locked deposit. If she behaves honestly, she can recover the deposit after the lock expires. However, if she equivocates, other principals (either an explicitly identified beneficiary or the cryptocurrency miners) can obtain her deposit instead. Ruffing *et al.* describe a construction of this protocol based on chameleon hash functions.

We take the work of Ruffing *et al.* as an exemplar for cryptocurrency-based mechanisms because it explicitly casts its goals in terms of accountability. The other mechanisms we review here are generally similar in their approaches and requirements. The mechanism of Ruffing *et al.* is focused on **punishment**. This punishment is mediated, as it requires either the beneficiary or the currency miners to construct information from the contradicting statements and use this information to redeem the deposit. However, we view this as very close to automatic punishment. This mechanism also provides some measure of **detection**, either when a payment is made early to a beneficiary or, probabilistically, in the case when payment is made by mining a block. Furthermore, these transactions redeeming the deposit implicitly provide some measure of **evidence** that is publicly available. Participants are required to have **identity** in the form of public keys. As part of detecting violations, **violations** are disclosed, and **violators** are identified as such, both broadly via the publicly visible transactions. This mechanism is **decentralized**, both with and without violations. The punishing **entity** is internal to the larger cryptocurrency system that uses this mechanism, and the punishment does not require **ongoing involvement** of the violator to punish her.

Ateniese, Goodrich, Lekakis, Papamanthou, Parakevas, and Tamassia (Ateniese *et al.*, 2017) describe a system for "accountable storage." Here, "accountability" is not formally defined, but it is taken to mean the assurance that an honest client will be paid in proportion to the amount of her data that is corrupted or lost. Earlier work on provable data possession or proofs of retrievability allowed a server to show that it still had the data stored on it by the client. Ateniese *et al.* consider the scenario in which a bounded amount of that data is corrupted and the client needs to be compensated. They describe a variation of their approach that is integrated with a cryptocurrency (in this case Bitcoin) that

> achieves one of the following outcomes within an established deadline. [This assumes that the server makes an initial "security deposit" of A bitcoins and that the measure of damage to the client's data is d.]

- If both the server and the client follow the protocol, the client gets exactly d bitcoins from the server and the server gets back his A bitcoins.
- If the server does not follow the protocol (*e.g.*, he tries to give fewer than d bitcoins to the client, fails to respond in a timely manner, or tries to forge [a] proof), the client gets A bitcoins from the server automatically.
- If the client requests more than d bitcoins from the server by providing invalid evidence, the server receives all A deposited bitcoins back and the client receives nothing.

The mechanism for doing this involves an arbitrator to resolve disputes (although Ateniese *et al.* discuss possible ways to remove this assumption), and it makes use of a Bitcoin transaction that is based on the work of Andrychowicz *et al.* (2014) noted above.

3.5 Summary of systems and mechanisms in computer science

Tables 3.1, 3.2, and 3.3 summarize our classification of the exemplar systems and mechanisms that we highlighted in the previous sections of this chapter. These are grouped according to whether they focus on internal evidence (or detection), providing evidence for third parties, making judgments or assigning blame, or meting out punishment. The leftmost column in each table identifies the mechanism. The columns, in Table 3.1, under the header "Time/Goals" identify which of prevention, detection, evidence, judgment/blame, and punishment are provided by the mechanism. For those mechanisms providing punishment, "Med." indicates that it is mediated punishment; for the other goals, a checkmark is used to indicate that the mechanism provides that goal. In all of these, parentheses around a table entry indicate that some additional qualification is given in the text in previous sections. This may be that the goals are not primary aims of the mechanism but that they are naturally achievable using the mechanism, that they are achieved to a limited degree or only under certain circumstances, or other qualifications.

The columns, in Table 3.2, under the header "Information" describe what level of identity is required for participation, whether and how broadly violations are disclosed, and whether and how broadly violators are identified as such. For identity requirements, the options are: "Pseud.," indicating that a pseudonym is required; "Host," indicating that hosts (but not users of those hosts) are identified; "Key," indicating that participants are identified by their public keys (although not *a priori* ruling out multiple public keys for an individual entity); "Unique," indicating that identity is known to a unique entity (*e.g.*, a trusted server); "Limited," indicating that identity is known to a well-defined, limited set of entities (*e.g.*, the intermediaries on a communication path); and "Broad," indicating that identity is known broadly and without prior constraints (*e.g.*, it is broadcast, or it is available to any node who wishes to look it up in a public database). Parentheses are used to indicate further qualification as described in the text above. The options for the disclosure of violations and the identification of violators as such are: "No/No," indicating that disclosure/identification is not made; "Unique/Unique," "Limited/Limited," and "Broad/Broad," indicating that disclosure/identification is made to a unique entity, limited set of entities, or broadly (as for the disclosure of identity). In some cases, multiple categorizations apply as discussed in the text above.

The columns, in Table 3.3, under the header "Action" describe whether the mechanism is centralized or decentralized (separate columns for with and without a violation occurring), whether there is a punishing entity involved, and whether punishment requires ongoing involvement by the violator. For centralization without/with a violation, the options are "Cent./Cent.," for centralized, and "Dec./Dec.," for decentralized. For a punishing entity, the entry can be "No," for no such entity, "Int.," for one that is internal to the mechanism (*e.g.*, one or more other participants), or "Ext.," for one that is assumed by the mechanism but is external to the mechanism (*e.g.*, the legal system). For ongoing involvement, the options are "No" and "Yes," for such involvement not required and required, respectively. In some cases, this can be ambiguous. In particular, in a mechanism that punishes violators by prohibiting their further participation in a system, it may be immaterial to distinguish a violator attempting to continue participating in the system (and being rejected) from a violator never again attempting to

	Time/Goals				
Approach/Paper	Prevention	Detection	Evidence	Judgment/Blame	Punishment
Internal Evidence (Sec. 3.1)					
AIP	(✓)	✓			(Med.)
APIP	(✓)	✓			(Med.)
PGPA		✓	(✓)		
Packet passports	(✓)		✓		(Med.)
AudIt/packet obit.		✓			
Evidence for Third Parties (Sec. 3.2)					
CATS		✓	✓	(✓)	
Accountable-subgroup multisig.	✓	✓	(✓)		
PeerReview & AVMs		✓	✓	(✓)	
Cryptographic commitments		✓	✓		
Time stamping		✓	✓		
Judgment or Blame (Sec. 3.3)					
DISSENT		✓	✓	✓	
Jagadeesan *et al.*, 2009				✓	
Barth *et al.*, 2007		✓	(✓)	✓	
Punishment (Sec. 3.4)					
A2SOCs			✓	✓	(Med.)
CHL off-line e-Cash		✓	✓	✓	Med.
B–LB Reputation				(✓)	Med.
PEREA					Med.
iOwe		✓	✓		Med.
Non-equivocation contracts		(✓)	(✓)		Med.

Table 3.1: Time-and-goals of accountability systems and mechanisms.

	Information		
Approach/Paper	Identity Requirements for Participation	Violation Disclosed?	Violator Identified as Such?
Internal Evidence (Sec. 3.1)			
AIP	Host	Broad	Broad
APIP	Unique	Broad	Unique
PGPA	(Unique)	Limited	Limited
Packet passports	Key	Limited	Limited
AudIt/packet obit.	Broad	Unique/Limited	Unique
Evidence for Third Parties (Sec. 3.2)			
CATS	Key	Broad	Broad
Accountable-subgp. multisig.	Broad	Unique/Broad	No/Broad
PeerReview & AVMs	Broad	Broad	Broad
Crypto. commitments			
Time stamping	Key	Limited	Limited
Judgment or Blame (Sec. 3.3)			
DISSENT	Key	Broad	Broad
Jagadeesan *et al.*, 2009	(Broad)	(Limited)	Unique
Barth *et al.*, 2007	(Broad)	Unique	Unique
Punishment (Sec. 3.4)			
A2SOCs	Unique	Broad	Broad
CHL off-line e-Cash	Key	Broad	Broad
B–LB Reputation	Broad	Broad	Broad
PEREA	Key	Unique	No
iOwe	Key	Broad	Broad
Non-equiv. contracts	Key	Broad	Broad

Table 3.2: Information classification of accountability systems and mechanisms.

	Action			
Approach/Paper	Centralization without Violation?	Centralization with Violation?	Punishing Entity?	Requires Ongoing Involvement?
Internal Evidence (Sec. 3.1)				
AIP	Dec.	Dec.	Int.	Yes
APIP	Dec.	Dec.	Int.	Yes
PGPA	Dec.	Dec.		
Packet passports	Dec.	Dec.	Int.	Yes
AudIt/packet obit.	Dec.	Dec.	Ext.	No
Evidence for Third Parties (Sec. 3.2)				
CATS	Dec.	Dec.	No	
Accountable-subgp. multisig.	Dec.	Dec.		
PeerReview & AVMs	Dec.	Dec.		
Crypto. commitments				
Time stamping	Cent.	Cent.		
Judgment or Blame (Sec. 3.3)				
DISSENT	Dec.	Dec.	No	
Jagadeesan *et al.*, 2009	Dec.	Dec.	No	No
Barth *et al.*, 2007	Cent.	Cent.	No	
Punishment (Sec. 3.4)				
A2SOCs	Cent.	Cent.	(Int.)	(Yes)
CHL off-line e-Cash	Cent.	Cent.	Int.	No
B–LB Reputation	Dec.	Dec.	Int.	Yes
PEREA	Dec.	Dec.	Int.	No
iOwe	Dec.	Dec.	Int.	Yes
Non-equiv. contracts	Dec.	Dec.	Int.	No

Table 3.3: Action classification of accountability systems and mechanisms.

participate in the system. However, we categorize such mechanisms as requiring ongoing participation because that is needed to realize the future non-participation as punishment.

3.6 Accountability mechanisms in other disciplines

To conclude this chapter, we turn to accountability mechanisms from outside of computer science. Decades of work in international relations and in public administration has considered "accountability." While the approaches in this section are not focused on computing, we find these perspectives helpful in sharpening thinking about accountability.

3.6.1 Accountability in international relations

Grant and Keohane (2005) explore the notion of accountability in international relations, although the mechanisms they identify are generally not limited to interactions between countries. Recall from Sec. 1.2 that they believe that accountability "implies that some actors have the right to hold other actors to a set of standards, to judge whether they have fulfilled their responsibilities in light of these standards, and to impose sanctions if they determine that these responsibilities have not been met." Their purpose in studying accountability mechanisms is that, in both international relations and national politics, such mechanisms can curb abuses of power.

Within democratic nations, abuse of power may be curbed by both preventive mechanisms and after-the-fact accountability mechanisms. In the US, preventive mechanisms are exemplified by checks and balances, such as presidential vetoes and legislative overrides of those vetoes, and accountability mechanisms are exemplified by elections. Elections are regarded as proper accountability mechanisms under the *participation model* of accountability, in which the performance of those who wield power is evaluated by those who are affected by their actions, and under the *delegation model*, in which it is evaluated by those who entrusted them with power. Unfortunately, this straightforward analysis does not carry over to international relations, because there are no trans-national, democratically elected governments.

Grant and Keohane identify seven accountability mechanisms that work with varying degrees of effectiveness in world politics. Four rely on delegation: hierarchy, supervision, funding, and law. For example, in a hierarchical international organization such as the United Nations, the head of the organization may delegate power to a subordinate official and can hold the subordinate official accountable for wielding that power appropriately through the possibility of demotion, firing, or loss of future career opportunities. Three of the mechanisms rely on participation: markets, peer-to-peer engagement, and public reputations. For example, in international markets, firms and governments wield power, and the market participants (consumers, lenders, stockholders, and bondholders) can hold them accountable through the threat of financial penalty (such as refusal to buy a firm's products or services, loss of access to capital, or higher cost of capital). Additional detail is given by Grant and Keohane (2005, Table 2).

Grant and Keohane also argue that international treaties between states in a balance-of-power system are not accountability mechanisms. Treaties might establish entities (*e.g.*, the United Nations) that are accountable, and they might establish "standards of legitimacy" to which states might be held. However, Grant and Keohane argue that treaties in which an offended state must take (*e.g.*, military) action on its own do not provide "accountability" even though they might disincentivize violations.

Weisband and Ebrahim (2007), in introducing their book, provide a review of accountability work outside of computer science. They note early work in the area dating back to the 1940s. Their review does not focus exclusively on international relations but does focus on "governance and social interaction."

3.6.2 Accountability in public administration

Koppell discusses different types of "accountability" in the context of public administration and identifies five views of it. For each, he identifies a related question that we quote here (see (Koppell, 2005, Table 1)):

Transparency Did the organization reveal the facts of its performance?

Liability Did the organization face consequences for its performance?

Controllability Did the organization do what the principal (*e.g.*, Congress, president) desired?

Responsibility Did the organization follow the rules?

Responsiveness Did the organization fulfill the substantive expectation (demand/need)?

Koppell gives a variety of examples of enforcement mechanisms for these types of accountability. For transparency, enforcement mechanisms include laws allowing access to governmental documents or meetings. Another example is required disclosures for public companies. For liability, one mechanism is elections and the need for elected officials to run for re-election; these are often held up as an example of an "accountable" system. In Sec. 4.3.2 we discuss work by Maskin and Tirole to model the incentives for officials who do and do not need to run for re-election.

Controllability might be exhibited in delegation settings, such as the United Nations example noted above. Koppell also cites arguments for a view that citizens control bureaucracy through their elected representatives, at least insofar as the bureaucracy does what they want it to. Notions of control appear prominent in the public-administration literature. Koppell points to work of Romzek and her collaborators (*e.g.*, (Radin and Romzek, 1996)); we see it also, *e.g.*, in a survey and taxonomy by Lindberg (2013).

In Koppell's typology, the rules that should be followed in order to demonstrate responsibility might be laws or they could be professional norms. For the fifth type, Koppell identifies responsiveness with meeting either the needs or demands of those being served (and contrasts this with other uses of "responsiveness" that connote a sense of controllability).

Writing earlier than Koppell, Mulgan (2000) surveyed and argued against expansions of "accountability" of the sort captured by Koppell's typology. As noted in Def. 2.8 above, Mulgan viewed the "core"

of accountability as "being called 'to account' to some authority for one's actions[.]" He articulated possible boundaries between "accountability" and "responsibility," while tracing the relationship between these. Mulgan also discussed internal notions of accountability, such as professional responsibility, in contrast to the external nature of the core principle he identified. The other expansions Mulgan argued against were accountability as "control," "responsiveness," and "dialogue" (in these sense of, *e.g.*, between citizens in a democratic society). We do not argue for a particular scope of "accountability" in public administration, but Mulgan's review provides a useful survey of the expansion of "accountability" into these areas and arguments for and against including them under the umbrella of "accountability."

Notes

Ko *et al.* (1993) provide an early approach to accountability through associating principals with their actions. They consider the problem of attacks that might seem innocuous on a per-host level and propose a system to observe users' movement across the network. Their focus is on how to algorithmically link actions—possibly under different names or on different hosts—to a single identity. This contrasts with other approaches that focus on cryptographic aspects of the problem such as ensuring the integrity of audit logs.

In Sec. 4.2.1, we discuss analysis work by Bella and Paulson (2006). They applied this to two protocols that connect actions to identities and provide evidence that can be presented to a third party (we discuss their view of accountability in Sec. 2.2.2). One of these protocols is a non-repudiation protocol of Zhou and Gollman (1996), and the other is a certified-email protocol of Abadi, Glew, Horne, and Pinkas (Abadi *et al.*, 2002).

Digital signatures that allow multiple potential signers raise various issues related to accountability. As one example of work in this area, Micali, Ohta, and Reyzin (Micali *et al.*, 2001) explicitly consider accountability as a goal in their "accountable-subgroup multisignatures." In particular, their goal is that "without use of trusted third parties, individual signers can be identified from the signed document."

Michalakis, Soulé, and Grimm (Michalakis *et al.*, 2007) present and test a mechanism called Repeat and Compare to ensure content integrity in peer-to-peer content-distribution networks. As they note, there are some similarities to the approach of PeerReview (Sec. 3.2.2); a notable contrast is that Repeat and Compare can handle some nondeterminism.

Liu, Xiao, and Gao (Liu *et al.*, 2011) leverage PeerReview and Kudo's extension of Kailar's logic (Sec. 2.2.2) to describe an approach to accountability in smart grids and sketch an analysis of this. They identify three accountability goals in this context: smart meters proving correctness of smart appliances, smart appliances proving correctness of smart meters, and service providers proving correctness of smart meters.

4

Reasoning About Accountability

Having discussed conceptions of accountability and mechanisms for providing accountability, we now turn to the study of accountability in systems. This includes work from a wide range of subdisciplines. We start in Sec. 4.1 with work on the development of tools—such as languages and formal logics—for use in reasoning about accountability-related properties. In Sec. 4.2 we review the results of other formal analyses of accountability. The focus of the work discussed in Sec. 4.2 is more on the properties proved than on the development of the tools used to prove them, but the approaches discussed there also involve the development of tools for reasoning about accountability that should be applicable to other systems.

We then turn, in Sec. 4.3, to accountability as a subject of study, including what can and cannot be proved about accountability in different settings, the relationship of accountability properties to incentives in different models, work to provide accountability while reducing other information that is needed (*e.g.*, to preserve privacy), and then some approaches to quantifying accountability.

4.1 Tools for reasoning about accountability

We start by considering tools that can be used to reason about, model, or analyze the accountability-related properties of systems. Most of the tools we highlight here are languages, which capture relevant properties and behavior, and formal logics, which might be used to derive or prove certain properties of a system. Someone, such as a system designer, might employ these to help make a formal argument that a particular system provides a desired property. These tools are typically focused on specific properties or application domains; for example, a system designer who wants to provide proofs of causality and blame would use a different tool than one who wants to analyze system logs to extract evidence of violations. We group these tools as relating to detection and audit, evidence, blame and causality, and other approaches. Many of these tools are used entirely by hand, while a few can be used with automated provers.

4.1.1 Tools related to detection and audit

Authorization logic for auditing

Vaughan, Jia, Mazurak, and Zdancewic (Vaughan *et al.*, 2008) present an authorization logic, AURA$_0$, that focuses on auditing. They "argue that audit[-]log entries should constitute *evidence* that justifies the authorization decisions made during the system's execution." Vaughan *et al.*'s approach involves a trusted kernel that mediates access to resources. In order to grant access to a resource, the kernel requires a proof that the access is allowed. The kernel then returns a proof that the operation was performed. The kernel also logs "the reasoning used to reach access[-]control decisions; if a particular access[-]control decision violates policy intent but is allowed by the rules, audit can reveal which rules contributed to this failure." Vaughan *et al.* consider three levels of logging—no logging, operation names without proofs (although proofs are required for access control), and full logging of proofs—in order to study what, and how much, information must be logged. In particular, they discuss how different levels of logging provide different information to auditors and how more information can allow more precise assignment of blame.

AURA$_0$ is cut down from a larger language to focus on auditing. Both of these languages are based on Abadi's Dependency Core Calculus (Abadi, 2007). In AURA$_0$, "types correspond to propositions relating to access control, and expressions correspond to proofs of these expressions." The language connects principals to their statements about access control, and it captures delegation at varying levels of generality. For example, Alice may say that Bob speaks for her, or she may say that Bob speaks for her with respect to a certain class of statements.

Algorithmically checking partial logs against policies

Garg, Jia, and Datta (Garg *et al.*, 2011) present an algorithm for checking partial logs for compliance with policies expressed in a first-order logic with quantifications. To ensure termination of their procedure, quantifications have restrictions that may be satisfied by only a finite number of variable substitutions. Their logic can capture "all 84 disclosure-related clauses of the HIPAA Privacy Rule[.]" Garg *et al.*'s algorithm works with incomplete logs; if it cannot be proved that a policy is satisfied, the algorithm produces a modified policy that captures the parts for which additional evidence is needed. This approach is extended by Oh, Chun, Jia, Garg, Gunter, and Datta (Oh *et al.*, 2014) to produce explanations for why a policy was or was not satisfied.

4.1.2 Tools related to evidence

Logics for third-party evidence

Kailar (1996) defined a proof logic for reasoning about accountability in the sense of Def. 2.3 above, *i.e.*, proving to third parties the association between a principal and data or actions. Accountability goals in a protocol might include enabling a customer to prove that a business agreed to sell a particular item at a particular price or enabling a business to prove that it provided a particular item to a customer. Once these goals are formalized for a particular protocol, and the message contents are formalized, this logic can be used to derive information about who can prove what to whom. This information can then be compared to the original accountability goals.

Kailar's logic includes a CanProve statement. In analyzing the accountability properties of a protocol, the goal is typically to derive a collection of CanProve statements. The logic includes a rule that allows the derivation of

$$A \text{ CanProve } (B \text{ Says } x)$$

—*i.e.*, that principal A can prove that principal B says statement x— from the receipt of a message containing x that is signed with a key that A can prove authenticates B. Explicitly framing this in terms of accountability, Kailar describes this rule as capturing that "a principal A can prove that another principal B is accountable for a statement which B has signed, if A can present B's key certificate (which is issued by some trusted authority, and hence, prove that B's signature can be authenticated by K" (Kailar, 1996, p. 318). Kailar applied this logic to protocols for electronic commerce and public-key delegation. As noted in Sec. 2.2.2, Kudo (1998) extends Kailar's logic to capture additional temporal considerations.

As noted by Kessler and Neumann (1998), Kailar does not provide a formal semantics for his logic. They extend an existing logic of belief with formal semantics to include a CanProve predicate and then prove the soundness of this extension. In applying their logic to e-commerce protocols, Kessler and Neumann do not provide an explicit definition of "accountability." They proceed roughly along the lines of Kailar, although they incorporate belief into their approach. Their formal goals are to prove, for a vendor V, customer C, judge J, and e-commerce order o, the statements

$$V \text{ Believes } C \text{ Said } o$$

and

$$V \text{ Believes } V \text{ CanProve } (C \text{ Said } o) \text{ to } J.$$

Here, Believes, CanProve . . . to, and Said are named to capture their intended roles.

Kungpisdan and Permpoontanalarp (2002) continue in this line of work. They explicitly consider information that should not be revealed to the third party, and they separately consider accountability for money and accountability for goods purchased.

Language and logic for data policies

In work of Corin, Etalle, den Hartog, Lenzini, and Staicu (Corin *et al.*, 2005), communications between agents are taken to provide evidence (assuming non-repudiable communication) that can be used to respond to external audits that examine actions and data usage. Corin *et al.* describe a language that captures abstract data-usage policies, ownership of data, and the communication/delegation of policies between agents (such as Alice telling Bob that he may read certain data). They present a logic for reasoning about policies, and they define two types of accountability in terms of agents being able to present, to an outside party, proofs in this logic about the possession and communication of policies.

Corin *et al.*'s notion of *agent accountability* captures the fact that, based on what an agent has been told, she is allowed to possess all policies that she possesses, and, at the time she sent a policy to another agent, she was allowed (by some other policy) to send that policy. Their notion of *data accountability* also looks recursively at the communications upon which these proofs rely; there must be proofs that the other agent(s) who delegated the policies needed for the proof were allowed to do so. (Corin *et al.* also consider a stronger version of data accountability.)

Cederquist, Corin, Dekker, Etalle, and den Hartog (Cederquist *et al.*, 2005) extend the work of Corin *et al.* by enriching the policy language and logic in multiple ways, including the addition of variables and quantifiers and the inclusion of use-once conditions. They also explicitly model logging, so that evidence available for proofs is not taken to be all communications but only those that are specifically logged; this also captures environmental information that may be needed for later proofs during an audit. They also mechanize their approach using the Twelf tool. Cederquist, Corin, Dekker, Etalle, den Hartog, and Lenzini (Cederquist *et al.*, 2007) present additional detail on this approach. In particular, they give a formal proof system and prove cut elimination for their logic, showing that it is semidecidable. They present a proof finder in Prolog that can be combined with the proof checker in Twelf. The expectation is that audited users would use the proof finder to produce proofs (perhaps before taking an action, to ensure that they

log sufficient information) that they can present to an auditor, who would run the checker on them.

4.1.3 Tools related to blame

A language for programs as actual causes

Datta, Garg, Kaynar, Sharma, and Sinha (Datta *et al.*, 2015) define a concurrent language for identifying programs as actual causes of violations. (As noted in Sec. 2.2.5, their approach to accountability focuses on blame.) The actions captured by their language include sending and receiving messages between threads and computing primitive functions, which are intended to model, *e.g.*, cryptographic operations. Datta *et al.* consider traces of this language and focus on safety properties, *i.e.*, properties that are described as sets of traces avoiding a finite set of violating prefixes.

Datta *et al.* define two types of causes: Lamport causes and actual causes. Informally, "[a] Lamport cause is a minimal projected prefix of the log of a violating trace that can account for the violation." An actual cause is a set of actions obtained from a Lamport cause by ignoring actions in the trace that serve only to enable the progress of the trace. Both types of causes satisfy sufficiency (for ensuring violations) and minimality conditions. Datta *et al.* present a procedure that they prove (Datta *et al.*, 2015, Theorem 1) will (1) find a Lamport cause for every violation, and, (2) for each Lamport cause, will find at least one actual cause of the violation. This applies to safety properties whose violations are "reordering closed" in the language of Datta *et al.*, *i.e.*, interchanging the order of actions in different threads does not affect the presence or absence of a violation. Notably, there may be multiple actual causes identified by this approach.

Datta *et al.* argue that it is important to distinguish between joint causes (sets of actions that cause a violation even if no subset causes a violation) and independent causes (different sets of actions that cause a violation, regardless of whether the other sets occur). If there are multiple actual causes found for a violation, these are independent.

Identifying agents who do not fulfill responsibilities

Barth *et al.* (2007), in their work described in Sec. 3.3.1, give an algorithm for identifying accountable agents. As noted in Def. 2.15 above, they view an agent as accountable for a violation if she undertakes a possible cause of the violation and did not fulfill her responsibilities. Their algorithm uses the logic for reasoning about privacy and utility that they present. This logic builds on linear temporal logic (for capturing privacy) and alternating-time temporal logic (for capturing utility).

Barth *et al.*'s approach relies on an oracle—expected to be a human monitor—to determine whether messages in the system are properly tagged. They prove that, if there is a violation, their algorithm is able to identify a participant who is accountable. While this participant is not guaranteed to have committed the actual cause of the policy violation, a human could check whether this is the case; if the participant did not commit the actual cause, the algorithm could be continued. Iterating this process would eventually enumerate all accountable agents, and the human monitor would be able to identify the correct one. Barth *et al.* also describe a heuristic for identifying suspicious actions and show that the existence of a suspicious action means that there is an incorrectly tagged message.

Automated analysis of accountability

Künnemann, Esiyok, and Backes (Künnemann *et al.*, 2018a) adapt an approach of Künnemann, Garg, and Backes (Künnemann *et al.*, 2018b) to the SAPiC extension of the applied π-calculus and then use the Tamarin prover to analyze a number of toy protocols as well as one for accountable algorithms presented by Kroll in his thesis (see Sec. 3.2.3).

The accountability goal of Künnemann, Esiyok, and Backes is to identify the protocol participant(s) who cause a violation. They

> define accountability as the ability of a protocol to point to any party that causes failure [with respect to] a security property φ. If there is no failure, no party should be accused. Hence accountability [with respect to] φ also implies verifia-

> bility of φ [citation omitted]. In this sense, accountability is a meta-property: we always talk about accountability for violations of φ (Künnemann *et al.*, 2018a, p. 3).

Künnemann, Esiyok, and Backes are interested in finding all minimal sets of parties whose deviations from the protocol are sufficient causes for the security violation in question. This depends on the way in which counterfactual traces are chosen; Künnemann, Esiyok, and Backes consider multiple ways in which to do that.

Künnemann, Esiyok, and Backes are able to define a condition that is necessary and sufficient for a verdict function they define to correctly compute all sets of parties who cause a violation. This condition can then be checked using software tools; they do this using the Tamarin prover.

4.1.4 Other approaches

Language for checking data usage across our temporal spectrum

Benghabrit, Grall, Royer, and Sellami (Benghabrit *et al.*, 2015) describe an approach to checking that users' preferences and requirements on the usage of their data are satisfied. They put this in the context of international requirements on data, with one goal being that "if [...] data is accepted to circulate in the system then the user preferences will be never violated, that is the accountability contract expected by the data subject is always satisfied." Benghabrit *et al.* take the view that "accountability [...] is the property of an entity being legally responsible for its acts[.]" They adopt the temporal spectrum from our work with Xiao (Feigenbaum *et al.*, 2012), reviewed in Sec. 2.1.1 above, as a guide for ensuring this, and they capture a coarsening of these stages in the Abstract Accountability Language (AAL) that they present.

Benghabrit *et al.* construct clauses in AAL that capture usage (prevention in our spectrum), audit (detection, evidence collection, and judgment in our spectrum), and rectification (incorporating punishment from our spectrum as well as compensation). Clauses are "contract[s] regarding usage, audit, and rectification expected for [...] data. A data subject (owner) agrees on such a contract for his data, expecting that this

contract will be ensured by the data controller and any data processor processing the data." Informally, these mean "[i]n any execution state, do the best to ensure the usage expression [...], and if a violation of the usage is observed by an audit [...] then the rectification applies" (Benghabrit *et al.*, 2015, p. 215). Clauses can be compared to ensure that the expected contract is satisfied even as data move to different holders, *etc.*

Benghabrit *et al.* show how to translate AAL expressions into first-order linear temporal logic. They then illustrate the use of a theorem prover (TSPASS) to check the satisfiability and validity of various accountability-related properties. This approach has been incorporated into the AccLab software project developed by Benghabrit in collaboration with Royer, Grall, Sellami, Teilhard, Tong, and Spens (Benghabrit *et al.*, n.d.; Benghabrit *et al.*, 2018).

A model for contracts

Zou, Wang, and Lin (Zou *et al.*, 2010) describe a model for contracts between software-as-a-service and cloud-service providers and their customers; these capture parties' obligations in ways that are amenable to automated analysis, and this approach is compatible with exisiting tools for logical analysis. Zou *et al.* do this work with a view of accountability as "clear disclosure of service obligations; faithfully honoring of disclosed obligations, or otherwise assuming the liability for the non-performance of the obligations." Zou *et al.* build on OWL-DL, a sublanguage of the Web Ontology Language (OWL), and the Semantic Web Rule Language (SWRL) for different aspects of their model. They can translate questions in their language to questions about Coloured Petri Nets and then apply existing software tools, developed for the latter, to aid in analysis.

Middleware to enforce accountability

Lin, Zou, and Wang (Lin *et al.*, 2010) survey accountability concepts in service-oriented settings and also provide an overview of their Llama system. Llama works as middleware between users and service processes to enforce a type of accountability. Lin *et al.* use a definition

of accountability, due to Schedler, Diamond, and Plattner (Schedler *et al.*, 1999), in which "A is accountable to B when A is obliged to inform B about A's (past or future) actions and decisions, or justify them and to be punished in the case of misconduct." Llama "provides the mechanisms for: (1) service[-]outcome monitoring; (2) probabilistic diagnosis to identify the likely cause of a problem; and (3) a reputation mechanism for preventing future problems."

4.2 Proofs about evidence in protocols

We now turn to results from analyses of accountability-related properties in protocols. The efforts that we review here all focus on proving properties about evidence. For evidence-focused accountability mechanisms, key issues are whether enough, and the right kind of, evidence has been collected in order to meet accountability goals. The efforts discussed in this section all involve some type of demonstration that the evidence produced in a protocol is sufficient for its intended accountability-related purpose.

Some of the formalization work discussed in Sec. 4.1 involves analysis, and the analyses we discuss in this section involve formalization in order to carry out the proofs. However, we view the work surveyed in this section as focused on proving that certain accountability properties are satisfied rather than on introducing new languages or logics.

4.2.1 Proving delivery of evidence in multiple types of protocols

As noted in Sec. 2.2.2, Bella and Paulson (2006) consider the delivery to a principal of evidence, for presentation to a third party, of another principal's participation in a protocol. Their general approach to accountability requires interpretation in the context of an individual protocol. As examples, they analyzed both a nonrepudiation protocol and a certified-email protocol. In the nonrepudiation case, the evidence provided to each party was cryptographically signed data from the other participant and a trusted third party. In the certified-email case, the sender gets a receipt and the recipient gets the email. This needs to be further refined to account for the reasoning power of the participants—

one might be interested in what the attacker can learn from all traffic she has seen, requiring potentially significant computation, while wanting an honest participant to receive evidence without needing to do computation not specified by the protocol. In the context of the email protocol, Bella and Paulson capture this as saying that, "if the (untrusted) sender [or receiver, respectively] *can derive* the return receipt [or email, respectively, ...] then the receiver [or sender, respectively] *has been given* the email [or return receipt, respectively]. [emphasis added]"

Bella and Paulson use the Isabelle theorem prover and their inductive method to prove theorems that formalize the validity of evidence as well as fairness in both protocols they consider.

4.2.2 Proving the existence of evidence in contract-signing protocols

Backes, Datta, Derek, Mitchell, and Turuani (Backes *et al.*, 2006) used a protocol logic, similar to one originally used for authentication, to prove properties of contract-signing protocols. One of these properties was accountability, which they defined as follows:

> Accountability means that if one of the parties gets cheated as a result of [the trusted third party] $\hat{T}$'s misbehavior, it will be able to hold $\hat{T}$ accountable. More precisely, at the end of every run where an agent gets cheated, its trace together with a contract of the other party should provide non-repudiable evidence that $\hat{T}$ misbehaved.

Backes *et al.* give the example of terms that can be used, in the logic they define, to derive a term that captures the dishonesty of the trusted third party. Thus, possessing these terms allows a cheated agent to prove misbehavior of the trusted third party. They then prove their evidence-based accountability property for the Asokan–Shoup–Waidner contract-signing protocol. They also argue that this holds for the Garay–Jakobsson–MacKenzie contract-signing protocol, although they omit the proof in that case.

The approach of Backes *et al.* makes use of proof "templates" that are applicable to other protocols with similar structures. It is also compositional in that in is built up from proofs of subprotocols.

4.2.3 Proving sufficient evidence for audit

Guts, Fournet, and Zappa Nardelli (Guts *et al.*, 2009) present an approach, implemented in ML, of verifying through typechecking that enough evidence is logged in order to achieve an audit goal. Their approach contrasts with those of Bella and Paulson and of Backes *et al.* in that Guts *et al.* work with concrete implementations of protocols in F# rather than with the protocol design.

Guts *et al.* start with the following definition:

Definition 4.1 (Protocol auditability (Guts *et al.*, 2009))**.** "[A] protocol is *auditable* with respect to a property if it logs enough evidence to convince an *impartial* third party, called a *judge*, of that property.

They present the analysis of a protocol for distributed games between multiple parties. They write this protocol in the F# dialect of ML and then use the F7 tool to prove that it is auditable in the sense that "once a player wins a game [...], it can reliably convince all other principals of his victory (according to a 'judge' procedure [...])." Guts *et al.* prove other, non-auditability properties of this protocol. They have also applied their approach to other protocols.

4.3 Accountability as a subject of study

We now consider various lines of work that provide results, in some form, about accountability as the subject. These include, *e.g.*, describing the extent to which accountability-related properties can or cannot be achieved in certain classes of systems. These also include approaches to modeling accountable systems that allow for formal arguments to be made about the requirements—on information or flow of utility—for accountability properties or the effects of accountability on other behavior.

4.3.1 Complexity, possibility, and impossibility results

Checking and restoring accountability in systems with obligations

As discussed in Sec. 2.2.5, Irwin, Yu, and Winsborough (Irwin *et al.*, 2006) define a notion of accountability when studying systems with

positive obligations. For example, a user might be granted access to data, but they would have the obligation to record their use of the data in a log within 24 hours of access.

Irwin *et al.* study strong and weak versions of their accountability property. These arise from multiple models for ensuring that the action a system participant is obliged to do in a specified time interval is in fact enabled by the system. In the strong version of Irwin *et al.*'s accountability, the action to satisfy an obligation is enabled throughout the time interval in which is must be performed. In the weak version, the action need only be enabled at the end of the time interval. Irwin *et al.* explicitly do not address an even weaker, existential version that would only guarantee the action is enabled at some time within the interval; such a guarantee seems unhelpful for participants with autonomy in their scheduling.

Irwin *et al.* study what they call the "*accountability problem*, *i.e.*, given a state in a system, determining whether it is accountable" (in the sense we discuss in Sec. 2.2.5). They prove that, under three certain restrictive conditions, this can be done in polynomial time for both strong and weak accountability. They also show that, in general, if a system satisfies only two of their three conditions, then the accountability problem is co-NP hard. Finally, they present a "concrete model" in which their conditions are relaxed but the accountability problem is still in P.

In subsequent work, Pontual, Chowdhury, Winsborough, Yu, and Irwin (Pontual *et al.*, 2011) focus on strong accountability and discuss, *inter alia*, ways in which their sense of accountability might be restored to a system. Possible techniques include removing, rescheduling, reassigning, or adding obligations. Chowdhury, Pontual, Winsborough, Yu, Irwin, and Niu (Chowdhury *et al.*, 2012) also focus on the strong sense of accountability defined by Irwin *et al.* They update the model to facilitate the use of "cascading" obligations—*i.e.*, obligations that are satisfied by actions that incur new obligations—and they study the effects of these obligations. They prove that, in general, the accountability problem is NP hard, but that accountability can be decided in polynomial time (as a function of various relevant sizes) for certain classes of cascading obligations.

Identifiable abort

There has been recent work on secure multiparty computation with *identifiable abort*. As defined by Ishai *et al.* (2014), this "allows the computation to fail (abort), but [it] ensures that when this happens every party is informed about it, and they also agree on the index i of some corrupted party[.]" Secure multiparty computation with identifiable abort provides a notion of accountability by identifying policy violators. Ishai *et al.* prove feasibility results for this property that complement earlier impossibility results.

Cohen and Lindell (2017) study the relationship between fairness (honest parties receiving output if, and only if, dishonest ones do) and guaranteed output delivery (all parties being guaranteed to receive output). As part of their work, they prove that

1. "[I]f a functionality can be securely computed with fairness and identifiable abort [...], then the functionality can be securely computed with guaranteed output delivery[;]" and

2. "[S]ecurity with identifiable abort cannot be achieved in general for [at least a third of the participants corrupted] without broadcast."

From our perspective, these results of Cohen and Lindell show what is provided by this accountability property—*i.e.*, identifiable abort—and limits on how it can be achieved.

Modeling and analyzing audit systems

As noted in Sec. 3.3.1, Jagadeesan *et al.* (2009) study theoretical issues surrounding audits. With an eye towards applying model-checking techniques, they model auditors in finitary distributed systems. As in other work, auditors rely on evidence—they do not get to see messages that are exchanged only between violating nodes but instead see what is recorded in audit logs that they receive. Additionally, they are confined to a run of the system and so are unable to check properties of sets of traces.

Jagadeesan *et al.* consider different goals for audit systems: blaming all guilty parties, blaming only guilty parties, *etc.* They prove various

results about the extent to which these are achievable, both in absolute terms and in exponentially bounding the complexity of the corresponding model-checking problems.

4.3.2 Accountability and incentives

Accountability requirements for incentivizing network innovation

Laskowski and Chuang (2006) incorporate "accountability" into a game-theoretic study of innovation in networks. They take "poor accountability" in the Internet to mean that the network "reveals little information about the behavior—or misbehavior—of ISPs." While this is focused on information, they incorporate this into a model that might economically punish ISPs that deviate from their recommendations. Laskowski and Chuang are interested in what network characteristics would foster innovation, which they take to mean investment by an ISP that increases the utility of at least one potential path in the network.

Laskowski and Chuang view contracts between ISPs as algorithms that take "proofs"—*e.g.*, of path quality—as inputs and compute payments to be made between ISPs. In their model, they show that, in essence, incentivizing innovation requires the use of a "contractible rest-of-path" monitor. Here, "contractible" means that it generates proofs suitable for use in contracts, *i.e.*, "a court[]

1. Can verify the monitor's proofs.

2. Can understand what the proofs and contracts represent to the extent required to police illegal activity.

3. Can enforce payments among contracting parties."

A "rest-of-path" monitor is one that "informs each node along the data path what the quality is for the rest of the path to the destination." Laskowski and Chuang (2006, Claim 3) show that contractible rest-of-path monitors allow them to achieve certain goals in an equilibrium notion that does not "rely on re-routing for punishment, is coalition proof, and [does not] punish innocent nodes when a coalition cheats."

Audit games

Blocki, Christin, Datta, Procaccia, and Sinha (Blocki *et al.*, 2013) study *audit games* in which the defender first chooses a distribution over n targets to audit. With knowledge of this strategy, the attacker then chooses one of the n targets to attack. It is better for the defender to audit the attacked target than an unattacked target, and it is better for the attacker to attack an unaudited target than an audited one. The difference between this approach and earlier work on security games is that Blocki *et al.* also add a notion of costly punishment. The attacker chooses a punishment "rate" and pays a cost proportional to that rate regardless of the targets audited and attacked; the defender pays the rate as punishment exactly when the attacked target is the one audited.

The work of Blocki *et al.* is not explicitly framed in terms of accountability, although it builds on earlier work of Blocki, Christin, Datta, and Sinha (Blocki *et al.*, 2012a) that is explicitly motivated by accountable data governance. As described by Blocki *et al.* (2013), the focus of the earlier work "is on developing a detailed model and using it to predict observed audit practices in industry and the effect of public policy interventions on auditing practices. [The earlier work (Blocki *et al.*, 2012a) does] not present efficient algorithms for computing the optimal audit strategy. In contrast, [the follow-up work (Blocki *et al.*, 2013) uses] a more general and simpler model and present an efficient algorithm for computing an approximately optimal audit strategy."

This line of work stands out from general audit work because its utility-theoretic model explicitly includes the effects of punishment.

Datta (2014) gives a short, high-level survey of his work with various collaborators on privacy and accountability. The discussion of accountability includes the work on audit games just described as well as auditing over partial logs. Datta's overview places these lines of work in the context of a larger research program.

Utility-theoretic model of punishment

In some of our earlier work (Feigenbaum *et al.*, 2011), we argued for a punishment-focused view of "accountability" (cf. Def. 2.14 above) that did not require an explicit punishing entity. In support of this approach,

we formalized the notion of punishment using a utility-theoretic, trace-based view of system executions. In this model, participants with private utility functions do actions that form traces. If a violation occurs, the violating principal is punished if her utility is decreased, relative to what it would have been without the violation, either with or without an intervening action that is explicitly intended to punisher her. This can be done either in a probabilistic sense or using "typical" utilities. When there is a punishing action, this must be causally linked to the fact of the violation so that, *e.g.*, "bad luck" does not count as punishment.

Some of our more recent work (Feigenbaum *et al.*, 2014) notes that it may be better to connect punishment to blameworthiness instead of the fact of the violation. Addressing an issue raised by Datta, that work also argues that punishment should be "targeted" and not collective, *e.g.*, decreasing the utility of every participant in a system in response to a violation. In terms of the formal model, we consider in our more recent work the punishment of principals who take on identities ("nyms") and then do actions within a system. We argue that a natural translation of quasilinear utilities does not capture sufficiently the effects of reputation. Instead, we propose "utilities with linear transfer" in which a principal's utility can be written as a term that depends on the trace alone plus terms that capture the utility that each nym derives from the trace, each modified by the strength of the binding between the principal and the nym. This allows us to argue informally for the claim (Feigenbaum *et al.*, 2014, Claim 7.1) that a system that provides punishment must be able to punish the nym that was used to commit the violation and do so in a way that the effect on the principal using the nym outweighs any positive side effects.

Modeling effects of accountability on public policy

We briefly note work by Maskin and Tirole (2004), who study an economic model of the effects of accountability on policy making. They say that a public official is "accountable" if she must "run for reelection every so often." They then model direct democracy (in which citizens themselves make decisions based on what is popular), the use of accountable representatives who make decisions for the public, and the

use of non-accountable officials ("judges") who also make decisions for the public but without the need to run for reelection. Maskin and Tirole study the effects of accountability on incentivizing representatives to act in welfare-maximizing (as opposed to popular) ways. This assumes that representatives are more likely to know the welfare-maximizing action, but the general public may or may not learn this information before the next election. In the Maskin–Tirole models, the relative merits of different decision-making approaches depends, *inter alia*, on the balance between holding power and ensuring a legacy as motivations for representatives, the knowledge of the general public on the issues at hand, and the cost to officials of identifying the optimal action.

While this work is outside of 'accountable" systems in computing, it provides one example of how the effects of punishment might be modeled.

4.3.3 Accountability with reduced information

An important question in studying accountability is, What information is required to achieve the desired accountability-related goals? As noted elsewhere in this survey, some approaches explicitly equate accountability with binding the identities of actors to their actions in a public way. Other approaches raise the question of how much identity information is required in order to achieve specified accountability-related properties in certain systems. This can be generalized to how much information—about identity, the specific actions taken, *etc.*—is required in order to achieve various accountability goals. While this is an important question for future work, and has motivated some of our previous work (Feigenbaum *et al.*, 2011; Feigenbaum *et al.*, 2014), we note here some work that has focused on this area. In doing so, we focus on their approaches to information rather than where they fit in our temporal spectrum of Sec. 2.1.1.

In describing APIP (see Sec. 3.1.1 above), Naylor, Mukerjee, and Steenkiste (Naylor *et al.*, 2014) analyze the different roles of a source address in traffic. They identify five of these as: return address, sender's identity, use in error reporting, flow ID, and accountability. As noted above, APIP uses a separate accountability address in order to preserve

privacy as much as possible. Of note to us here is the identification of different roles played by the source address and the use of the separate accountability address in order to avoid conflating accountability and other goals.

Along with discussing the definition given in Def. 2.12 above, Lampson argues that accountability can be had in a system without forcing users to give up all of their privacy. In particular, he says that

> [a]ccountability is not the opposite of anonymity or the same as total loss of privacy. The degree of accountability is negotiated between the parties involved [. . .]; if there's no agreement, then nothing is disclosed and they stop interacting.
>
> [. . .]
>
> Accountability requires a consistent identifier based upon a name, a pseudonym or a set of attributes. When the identifier is based upon a name, the recipient may use a reputation service to determine whether the sender is accountable enough. Should the sender behave unacceptably, then the recipient can "punish" the sender by reducing the sender's reputation.
>
> When the identifier is a pseudonym, it must be issued by an indirection service which knows the true identity of the sender. When the sender behaves unacceptably, the indirection service may be requested to reveal the real-world identity to appropriate authorities by those authorities.
>
> [. . .]
>
> Becoming accountable does not necessarily mean disclosing anything about your real-world identity thus protecting privacy. (Lampson, 2005b, notes on slide 9)

While this suggests ways in which participants in a system might keep their identities from being broadly known, and it envisions negotiation about the level of accountability (and, implicitly, the extent to which a participant's identity will be disclosed) in order to participate in a system, this does connect identity to "accountability."

Another view of accountability while limiting disclosure by system participants is seen in the work of Kroll *et al.* (2017) discussed in Sec. 2.3.2. In the context of accountable algorithms, the goal of procedural regularity that they seek provides reduced information (through, *e.g.*, the use of zero-knowledge proofs) about the procedures (rather than the identities) used. They outline cryptographic approaches to provide their goal of procedural regularity. Kroll in his thesis (Kroll, 2015) described related technical details. Künnemann, Esiyok, and Backes (Künnemann *et al.*, 2018a) provide a formal analysis of an extension of a protocol presented by Kroll in his thesis. That extension addresses an attack on that protocol, not in the scope of Kroll's initial analysis, that was found by Künnemann, Gard, and Backes (Künnemann *et al.*, 2018b).

Notes

Sandhu and Samarati (1996) give an early survey of work on auditing. They identify two key aspects of audit systems: "the collection and organization of audit data, and an analysis of the data to discover or diagnose security violations."

Marinovic, Dulay, and Sloman (Marinovic *et al.*, 2014) present a language for break-glass policies. This models gaps in knowledge and potentially conflicting evidence when evaluating requests for exceptional access. Marinovic *et al.* envision decisions being made to grant exceptional access, grant it subject to additional conditions, or deny it. As an example, they give a HIPAA-compliant break-glass policy.

It is natural to ask how accountability-related properties might be measured. We note here some of the work that has been done on this. Nuñez, Fernandez-Gago, Pearson, and Felici (Nuñez *et al.*, 2013) describe a metamodel that can guide users toward the development of a model for measuring an accountability-related property of interest to the user. As the user develops a model, she identifies metrics, evidence used to compute the metrics, and other factors.

The report (Centre for Information Policy Leadership, 2010) of the Paris meeting of The Accountability Project considers measurement of accountability in the sense of that project (see Notes at end of Chap. 2). It presents "stages in the measurement of an organi[z]ation's

accountability program." Fitting with the focus of that project, these are degrees of organizational evaluation rather than quantitative measures of a system.

In approaches such as our utility-theoretic view of accountability, potential measurements of accountability-related properties may be sensitive to modeling assumptions. It may be helpful to instead compare systems in terms of a specific property. For example, we might ask, for systems A and B, whether A produces evidence of a violation whenever B produces evidence of that violation, and perhaps also whether the evidence in A at least as strong as that in B. Concretely, B might provide principal-generated logs of all interactions (as in PeerReview), while A might force communications to be over channels that generate trustworthy logs. A would produce evidence of violations for which B produces evidence, but it would also produce evidence of violations that involve only communications between two colluding parties.

5

Conclusions

As Chaps. 2, 3, and 4 clearly show, there has been a great deal of work on accountability or, more generally, on techniques that complement traditional preventive security mechanisms. Unfortunately, much of it has been done without knowledge of previous and concurrent work in the same area, and the result is a plethora of proposed techniques and mechanisms rather than a small set of approaches that are deployed and well understood. Nonetheless, we believe that a few key ideas stand out in the extensive but heretofore poorly organized work in this area.

In disciplines other than computer science (and even in CS areas most heavily influenced by other disciplines, *e.g.*, "accountable algorithms"), the term "accountability" often focuses on giving an account to another party. There are often requirements to give an account that is "satisfactory" to a third party or to participate in an interactive account in which the third party adaptively asks about past behavior. By contrast, computer-science uses of the term "accountability" have typically focused on detecting misbehavior, gathering evidence of misbehavior (in an automated way, as opposed to via the "giving of an account"), judging that behavior was a policy violation, or blaming a participant for a policy violation. In the language of the temporal

spectrum that we proposed in Chap. 2, most of these ideas are focused on the detection, evidence, and judgment-or-blame phases.

In this chapter, we summarize the key ideas in accountability that have been explored in depth, point to some key papers, and suggest directions for future work.

5.1 Summary of key ideas

Accountability through identification of bad actors

Several influential works equate accountability with identification of bad actors. Anonymity or pseudonymity is often a key feature of these systems: Participants who conform to all of the system policies can accomplish their goals without having to identify themselves to other participants; those who violate system policies will have their identities revealed, and the accountability mechanisms will provide irrefutable evidence of the violation. This approach was pioneered in the design of e-cash systems. For example, the "off-line" e-cash system of Chaum *et al.* (1990) does not prevent double spending but rather uses cryptographic techniques to ensure that double spending will be detected after the fact and that, when it is detected, the identity of the double spender is revealed to the bank. Camenisch *et al.* (2006) later extended this approach in a system that guarantees anonymity to any participant who conforms to a policy that both precludes double spending and imposes a spending limit but reveals the identities of violators. Accountability through (provable) identification of bad actors is also used in the DISSENT protocol (Corrigan-Gibbs and Ford, 2010; Syta *et al.*, 2014) for anonymous communication by closed groups. Each execution of the protocol includes a validation phase, in which the group attempts to prove to itself that previous phases were conducted according to policy; if validation fails, the protocol moves on to a blame phase, in which a violator is identified.

In other influential works, the term "accountability" is taken to mean the ability to identify the entity or entities who are responsible for an action, regardless of whether the action conforms to policy or is a violation. The accountable systems in these scenarios are usually network

protocols; anonymity is not a design goal, but, because the protocols were designed when networks were small and adversarial behavior was rare, authentication mechanisms are often lacking. Reliable association of an action with the participant(s) responsible for it thus requires the addition of an "accountability protocol." Examples include the early work of Ko *et al.* (1993) and the Accountable Internet Protocol (AIP) of Andersen *et al.* (2008).

Accountability through the identification of root causes

Closely related to the notion that "accountability" can be achieved by identifying bad actors is the notion that it can be achieved by identifying bad actions or "root causes." This approach is exemplified by the work of Barth *et al.* (2007) on privacy policies in business processes. In their model, violations of privacy policy can be caught at run time in the execution of a policy non-compliant workflow, and auditing algorithms can be used to identify a violating actor. The algorithms are not fully automatic but can reduce the amount of effort required of a human auditor to a point at which it is feasible.

Datta *et al.* (2015) explore the related idea of program actions as "actual causes" of a violation. They distinguish between actions that merely do not follow protocol and actions that *provably cause* a violating event; their work highlights the need to examine interactions among multiple agents in a decentralized system with non-deterministic execution semantics.

Irwin *et al.* (2006) study the interpretation of security policies from the point of view of *positive obligations*, *i.e.*, requirements that certain actions be taken whenever certain conditions hold. They define an *accountable state* of a system as one in which every participant is enabled to fulfill all of his obligations. Clearly, an unfulfilled obligation is a security-policy violation, but it is important to distinguish between an obligation that remains unfulfilled because necessary conditions for fulfilling it were not met (*e.g.*, because one or more other obligations were not fulfilled at an earlier stage of the execution) and one that remains unfulfilled because a participant that was able to fulfill it did not do so. Every violation v of this form can be traced back to a root

cause, *i.e.*, a participant who did not fulfill an obligation that he was enabled to fulfill, thus causing a sequence of events that led to v. Irwin *et al.* study the complexity of identifying root causes and show that the problem is tractable in many natural cases.

It is our view that identification (of bad actors or bad actions) is more accurately thought of as an *enabler* of accountability than as an accountability mechanism *per se*. Once a violation has been proven to have occurred and the violator identified, the system can impose consequences on the violator, *e.g.*, by fining him or punishing him in some quantifiable way that makes sense in context and is proportionate to the violation or by expelling him from the system entirely. It is punishment for policy violations that we view as "accountability," and, as noted in Chap. 3, identification is not necessarily a pre-requisite for punishment.

Accountability through the acquisition, preservation, and use of evidence

A key idea that appears in many otherwise disjoint works in the literature is that holding an entity A "accountable" means convincing another entity that A did (or, in some cases, did not) perform a certain action. The challenge is to ensure that sufficient evidence of all relevant actions is captured and maintained, that it cannot later be altered so as to make it useless or misleading, that it is delivered to the relevant parties before it is needed, and that the computational overhead of all of this activity by the accountability mechanism does not intolerably degrade the performance of the ambient system. Notable works that equate accountability with the gathering and use of evidence are those of Bella and Paulson (2006), Yumerefendi and Chase (Yumerefendi and Chase, 2004; Yumerefendi and Chase, 2005; Yumerefendi and Chase, 2007), and Haeberlen *et al.* (2007). In their work on PeerReview, Haeberlen *et al.* succinctly summarize this approach by defining an accountable system as one that "maintains a tamper-evident record that provides non-repudiable evidence of all nodes' actions."

Another line of research that was pursued contemporaneously with PeerReview (and other works that require tamper-evident logs and rigorous proofs of which actions were taken by whom) is exemplified

by the AudIt mechanism of Argyraki *et al.* (2007); we refer to this contemporaneous alternative as the *lightweight-localization* approach. As explained in Chap. 3, AudIt is similar to PeerReview in that it considers "evidence" to be central to the attainment of accountability in a distributed system. However, AudIt's lightweight localization is tailored to the Internet, in which the system participants are autonomous systems (ASes), among whom there are formal business relationships. In the context of contracts between business partners, it can be sufficient for an accountability mechanism to provide evidence, which may fall short of formal proof, to the affected ASes without adopting the responsibility to prove or even explain anything to the entire network.

The notion of "accountability" embodied in the lightweight-localization approach is "performance feedback [that helps] establish whether providers (and peers) are adequately performing their duty" (Argyraki *et al.*, 2007). Argyraki *et al.* show how it can be used to provide information to a source AS about where in the network its outgoing packets are dropped or delayed. The approach was anticipated in earlier work on "packet obituaries" (Argyraki *et al.*, 2004).

Focus on punishment and a terminological question

In Chap. 3, we discussed our own formulation of "accountability," which rests entirely upon the tying of policy violations to negative consequences. Unlike previous formulations, ours encompasses mechanisms that punish violators automatically, without identifying them to other participants in the system or engaging them in an adjudication procedure in which evidence is presented. Indeed, we do not even require that a violator be identified to himself; an agent who violates a policy and, as a consequence, experiences lower utility than he would have if he had obeyed it is said to have been held accountable. For example, a bidder in a strategyproof auction who lowballs his bid and does not win an item that he would have won if he had bid truthfully has, in our framework, been held accountable for not following the "bid your true value" rule of the auction system.

As explained in Chap. 2, our formulation has been praised for elevating the importance of tying violations to quantifiable consequences,

but it has been criticized for its use of the term "accountability." Critics suggest that the property we have formulated is more accurately referred to as "deterrence": If all participants know that there is a significant chance that they will suffer negative consequences if they violate policy (even if they may not learn exactly what these consequences are or when they were [automatically] imposed), then they will be deterred from violating. To be held "accountable," however, they must be told that they have committed a violation, and it must be clear to them and to others in the system exactly why the action for which they are being punished is a violation and why the punishment is just. This notion of accountability, in which violations are deterred because it is known that they may lead to negative consequences *and* there is a formal "accounting" of precisely which violation was committed by whom, how severely it is to be punished, and why, is best exemplified by the criminal justice system.

5.2 Key papers

In addition to those cited in Sec. 5.1, the following papers marked new and interesting directions in the study of accountability.

- To the best of our knowledge, Nissenbaum (1997) was the first to foreground the word "accountability" in the study of computer systems. She placed accountability into the discourse of "human values" in technology generally and computers and software in particular.

- More than a decade later, Weitzner *et al.* (2008) and Lampson (2009) brought the term "accountability" into a prominent position in the study of computer security and user privacy. The key contributions of these papers were a sober appraisal of the inadequacy of preventive security and privacy measures in Internet-scale applications and protocols and the suggestion that before-the-fact prevention could be combined with after-the-fact accountability, by analogy with offline, real-world security regimes.

- Our own work (Feigenbaum *et al.*, 2011), which followed Weitzner *et al.* (2008) by just a few years, shifted the focus to the last

point on the temporal of spectrum of Chap. 2, *i.e.*, to punishment. Our thesis is that "accountability" should be achieved by tying (violating) actions to consequences. By demonstrating that the tie between actions and consequences could be made without identification of violators or formal judgment-or-blame procedures, this work raises the question of whether "accountability" should be distinguished from "deterrence" (and perhaps from other categories of non-preventive security measures).

- Most recently, Frankle *et al.* (2018) and Kroll *et al.* (Kroll, 2015; Kroll *et al.*, 2017) have demonstrated the need for new accountability mechanisms in an era of bulk surveillance. The novel accountable-surveillance systems that they propose make heavy use of cryptographic-computing techniques.

- Outside of computer science, the papers of Grant and Keohane (2005), Mulgan (2000), and Raab (2012) contribute key ideas to the role of "accountability" in international relations, public administration, and data protection, respectively.

5.3 Future work

The most important agenda item, by far, for further work on accountability in computing systems is implementation, deployment, and evaluation of some of the apparently useful mechanisms that we have surveyed. Although there has been a great deal of solid research on accountability mechanisms, few if any of the novel mechanisms that appear in the literature have been deployed at scale. If they are not actually usable in their current form in large-scale, real-world systems, then they should be studied further and modified for use. There are clearly some useful ideas and techniques in the accountability literature; given that pure prevention has proven obviously sufficient, we believe that security professionals are obligated to add accountability mechanisms to their toolkits.

Social media platforms are potentially fertile ground for the roll out of online accountability. Facebook and Twitter are characterized by weak user identities and at-best limited roles for classic cryptographic

protocols. Accountability concepts such as detection, evidence, and punishment may be more effective than standard security concepts such as authentication and encryption.

Of course, there are also opportunities for further development of the theory of accountability in computing. For example, are there fundamental tradeoffs between accountability-related properties and privacy-related properties? Are there specific contexts in which precise statements can be made about the identity information that is provably required in order to achieve a certain notion of accountability?

Is it feasible to quantify and measure accountability properties that have heretofore been treated qualitatively? For example, can one measure the amount of deterrence provided by various accountability mechanisms as a function of the computational and operational costs of using those mechanisms?

Finally, future work on security and privacy protocols should address "accountability," explicitly identify desired accountability-related properties, and, whenever possible, prove that these properties are satisfied by the proposed protocols.

References

Abadi, M. 2007. "Access Control in a Core Calculus of Dependency". *Electron. Notes Theor. Comput. Sci.* 172(Apr.): 5–31. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2007.02.002.

Abadi, M., N. Glew, B. Horne, and B. Pinkas. 2002. "Certified email with a light on-line trusted third party: design and implementation". In: *Proceedings of the 11th international conference on World Wide Web. WWW '02*. ACM. 387–395. ISBN: 1-58113-449-5. DOI: 10.1145/511446.511497.

Aditya, P., M. Zhao, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, and B. Wishon. 2012. "Reliable Client Accounting for P2P-infrastructure Hybrids". In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. NSDI '12*. Accessed November 23, 2020. USENIX Association. 99–112. URL: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/aditya.

Andersen, D. G., H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. 2008. "Accountable Internet Protocol (AIP)". In: *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication. SIGCOMM '08*. ACM. 339–350. ISBN: 978-1-60558-175-0. DOI: 10.1145/1402958.1402997.

Andrychowicz, M., S. Dziembowski, D. Malinowski, and Ł. Mazurek. 2013. "Fair Two-Party Computations via Bitcoin Deposits". Cryptology ePrint Archive, Report 2013/837. https://eprint.iacr.org/2013/837.

Andrychowicz, M., S. Dziembowski, D. Malinowski, and Ł. Mazurek. 2014. "Secure Multiparty Computations on Bitcoin". In: *Proceedings of the 2014 IEEE Symposium on Security and Privacy. SP '14*. IEEE Computer Society. 443–458. ISBN: 978-1-4799-4686-0. DOI: 10.1109/SP.2014.35.

Andrychowicz, M., S. Dziembowski, D. Malinowski, and Ł. Mazurek. 2016. "Secure Multiparty Computations on Bitcoin". *Commun. ACM*. 59(4): 76–84. ISSN: 0001-0782. DOI: 10.1145/2896386.

Argyraki, K., P. Maniatis, D. Cheriton, and S. Shenker. 2004. "Providing Packet Obituaries". In: *Third Workshop on Hot Topics in Networks. HotNets III*. Accessed November 23, 2020. ACM. URL: http://www.icsi.berkeley.edu/pubs/networking/packetobituaries04.pdf.

Argyraki, K., P. Maniatis, O. Irzak, S. Ashish, and S. Shenker. 2007. "Loss and Delay Accountability for the Internet". In: *2007 IEEE International Conference on Network Protocols. ICNP '07*. IEEE. 194–205. DOI: 10.1109/ICNP.2007.4375850.

Ateniese, G., M. T. Goodrich, V. Lekakis, C. Papamanthou, E. Paraskevas, and R. Tamassia. 2017. "Accountable Storage". In: *Proceedings of the 15th International Conference on Applied Cryptography and Network Security. ACNS '17*. Springer International Publishing. 623–644. ISBN: 978-3-319-61204-1. DOI: 10.1007/978-3-319-61204-1_31.

Au, M. H., P. P. Tsang, and A. Kapadia. 2011. "PEREA: Practical TTP-free Revocation of Repeatedly Misbehaving Anonymous Users". *ACM Trans. Inf. Syst. Secur.* 14(4): 29:1–29:34. ISSN: 1094-9224. DOI: 10.1145/2043628.2043630.

Aumann, Y. and Y. Lindell. 2010. "Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries". *J. Cryptol.* 23(2): 281–343. ISSN: 0933-2790. DOI: 10.1007/s00145-009-9040-7.

Backes, M., J. Clark, A. Kate, M. Simeonovski, and P. Druschel. 2014. "BackRef: Accountability in Anonymous Communication Networks". In: *Proceedings of the 12th International Conference on Applied Cryptography and Network Security. ACNS '14*. Springer International Publishing. 380–400. ISBN: 978-3-319-07536-5. DOI: 10.1007/978-3-319-07536-5_23.

Backes, M., A. Datta, A. Derek, J. C. Mitchell, and M. Turuani. 2006. "Compositional Analysis of Contract-signing Protocols". *Theor. Comput. Sci.* 367(1): 33–56. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2006.08.039.

Backes, M., P. Druschel, A. Haeberlen, and D. Unruh. 2009. "CSAR: A Practical and Provable Technique to Make Randomized Systems Accountable". In: *Proceedings of the Network and Distributed System Security Symposium. NDSS '09*. Accessed November 23, 2020. The Internet Society. URL: https://www.ndss-symposium.org/ndss2009/csar-practical-and-provable-technique-make-randomized-systems-accountable/.

Barocas, S. and A. D. Selbst. 2016. "Big Data's Disparate Impact". *Calif. L. Rev.* 104(3): 671–732. DOI: 10.15779/Z38BG31.

Barth, A., A. Datta, J. Mitchell, and S. Sundaram. 2007. "Privacy and Utility in Business Processes". In: *Proceedings of the 20th IEEE Computer Security Foundations Symposium. CSF '07*. IEEE Computer Society. 279–294. ISBN: 0-7695-2819-8. DOI: 10.1109/CSF.2007.26.

Baum, C., E. Orsini, and P. Scholl. 2016. "Efficient Secure Multiparty Computation with Identifiable Abort". Cryptology ePrint Archive, Report 2016/187. https://eprint.iacr.org/2016/187.

Bella, G. and L. C. Paulson. 2006. "Accountability Protocols: Formalized and Verified". *ACM Trans. Inf. Syst. Secur.* 9(2): 138–161. ISSN: 1094-9224. DOI: 10.1145/1151414.1151416.

Benghabrit, W., H. Grall, J.-C. Royer, and M. Sellami. 2015. "Abstract Accountability Language: Translation, Compliance and Application". In: *2015 Asia-Pacific Software Engineering Conference. APSEC '15*. 214–221. DOI: 10.1109/APSEC.2015.14.

Benghabrit, W., J.-C. Royer, H. Grall, M. Sellami, P. Teilhard, A. Tong, and J. Spens. 2018. "AccLab GitHub page, version 2.3". Accessed August 20, 2020. URL: https://github.com/hkff/AccLab.

Benghabrit, W., J.-C. Royer, H. Grall, M. Sellami, P. Teilhard, A. Tong, and J. Spens. "AccLab homepage". Accessed August 23, 2018. URL: https://web.imt-atlantique.fr/x-info/acclab/.

Bentov, I. and R. Kumaresan. 2014a. "How to Use Bitcoin to Design Fair Protocols". Cryptology ePrint Archive, Report 2014/129. https://eprint.iacr.org/2014/129; shorter version published as Bentov and Kumaresan, 2014b.

Bentov, I. and R. Kumaresan. 2014b. "How to Use Bitcoin to Design Fair Protocols". In: *Advances in Cryptology: Proceedings of the 34th Annual Cryptology Conference, Part II. CRYPTO '14*. Springer. 421–439. ISBN: 978-3-662-44381-1. DOI: 10.1007/978-3-662-44381-1_24.

Blocki, J., N. Christin, A. Datta, A. D. Procaccia, and A. Sinha. 2013. "Audit Games". In: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. IJCAI '13*. Accessed November 23, 2020. AAAI Press. 41–47. ISBN: 978-1-57735-633-2. URL: https://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6951.

Blocki, J., N. Christin, A. Datta, and A. Sinha. 2012a. "Audit Mechanisms for Provable Risk Management and Accountable Data Governance". *Tech. rep.* No. CMU-CyLab-12-020. http://repository.cmu.edu/cgi/viewcontent.cgi?article=1108&context=cylab Accessed 27 February 2018. Shorter version published as Blocki, Christin, Datta, and Sinha, 2012b. CyLab, Carnegie Mellon University.

Blocki, J., N. Christin, A. Datta, and A. Sinha. 2012b. "Audit Mechanisms for Provable Risk Management and Accountable Data Governance". In: *Decision and Game Theory for Security. GameSec '12*. Springer. 38–59. ISBN: 978-3-642-34266-0. DOI: 10.1007/978-3-642-34266-0_3.

Bovens, M. 2007. "Analysing and Assessing Accountability: A Conceptual Framework". *European Law Journal*. 13(4): 447–468. ISSN: 1468-0386. DOI: 10.1111/j.1468-0386.2007.00378.x.

Brickell, J. and V. Shmatikov. 2006. "Efficient Anonymity-Preserving Data Collection". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '06*. ACM. 76–85. ISBN: 978-1-4503-1651-4. DOI: 10.1145/1150402.1150415.

Buchegger, S. and J.-Y. Le Boudec. 2003. "A Robust Reputation System for Mobile Ad-hoc Networks". *Tech. rep.* Shorter version published as Buchegger and Le Boudec, 2004. EPFL.

Buchegger, S. and J.-Y. Le Boudec. 2004. "A Robust Reputation System for Mobile Ad-hoc Networks". In: *Proceedings of the Third Workshop on Economics of Peer-to-Peer Systems. P2P Econ '04.*

Buldas, A., P. Laud, and H. Lipmaa. 2000a. "Accountable Certificate Management Using Undeniable Attestations". In: *Proceedings of the 7th ACM Conference on Computer and Communications Security. CCS '00.* ACM. 9–17. ISBN: 1-58113-203-4. DOI: 10.1145/352600.352604.

Buldas, A., P. Laud, H. Lipmaa, and J. Villemson. 1998. "Time-stamping with binary linking schemes". In: *Advances in Cryptology: Proceedings of the 18th Annual International Cryptology Conference. CRYPTO '98.* Springer. 486–501. ISBN: 978-3-540-68462-6. DOI: 10.1007/BFb0055749.

Buldas, A., H. Lipmaa, and B. Schoenmakers. 2000b. "Optimally Efficient Accountable Time-Stamping". In: *Public Key Cryptography: Third International Workshop on Practice and Theory in Public Key Cryptosystems. PKC '00.* Springer. 293–305. ISBN: 978-3-540-46588-1. DOI: 10.1007/978-3-540-46588-1_20.

Buttyán, L. and J.-P. Hubaux. 1999. "Accountable anonymous access to services in mobile communication systems". In: *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems.* IEEE Computer Society. 384–389. DOI: 10.1109/RELDIS.1999.805128.

Camenisch, J., S. Hohenberger, and A. Lysyanskaya. 2006. "Balancing Accountability and Privacy Using E-Cash (Extended Abstract)". In: *Security and Cryptography for Networks. SCN '06.* Springer. 141–155. DOI: 10.1007/11832072_10.

Camenisch, J. and A. Lysyanskaya. 2001. "An efficient system for non-transferable anonymous credentials with optional anonymity revocation". In: *Advances in Cryptology. EUROCRYPT '01.* Springer. 93–118. DOI: 10.1007/3-540-44987-6_7.

Camenisch, J., A. Lysyanskaya, and M. Meyerovich. 2007. "Endorsed e-cash". In: *Proceedings of the 28th IEEE Symposium on Security and Privacy. SP '07*. IEEE Computer Society. 101–115. DOI: 10.1109/SP.2007.15.

Cederquist, J. G., R. Corin, M. A. C. Dekker, S. Etalle, and J. I. den Hartog. 2005. "An Audit Logic for Accountability". In: *Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks. POLICY '05*. IEEE Computer Society. 34–43. ISBN: 0-7695-2265-3. DOI: 10.1109/POLICY.2005.5.

Cederquist, J. G., R. Corin, M. A. C. Dekker, S. Etalle, J. I. den Hartog, and G. Lenzini. 2007. "Audit-based compliance control". *Int. J. Inf. Secur.* 6(2): 133–151. ISSN: 1615-5270. DOI: 10.1007/s10207-007-0017-y.

Centre for Information Policy Leadership. 2009. "Data Protection Accountability: The Essential Elements–A Document for Discussion". Accessed 26 January 2018. URL: https://iapp.org/media/pdf/knowledge_center/Galway_Accountability.pdf.

Centre for Information Policy Leadership. 2010. "Demonstrating and Measuring Accountability: A Discussion Document". https://iapp.org/media/pdf/knowledge_center/Accountability_Phase_II.pdf, accessed 25 January 2018.

Cerf, V. G. and R. E. Kahn. 1974. "A Protocol for Packet Network Intercommunication". *IEEE Trans. Commun.* 22(5): 637–648. ISSN: 0090-6778. DOI: 10.1109/TCOM.1974.1092259.

Chandra, T. D. and S. Toueg. 1996. "Unreliable Failure Detectors for Reliable Distributed Systems". *J. ACM.* 43(2): 225–267. ISSN: 0004-5411. DOI: 10.1145/226643.226647.

Charlesworth, A. and S. Pearson. 2013. "Developing accountability-based solutions for data privacy in the cloud". *Innovation: Eur. J. Soc. Sci. Res.* 26(1–2). ISSN: 1351-1610. DOI: 10.1080/13511610.2013.732753.

Chaum, D., A. Fiat, and M. Naor. 1990. "Untraceable electronic cash". In: *Advances in Cryptology. CRYPTO '88*. Springer-Verlag New York. 319–327. ISBN: 0-387-97196-3. DOI: 10.1007/0-387-34799-2_25.

Chaum, D. 1982. "Blind signatures for untraceable payments". In: *Advances in Cryptology. CRYPTO '82*. Springer. 199–203. DOI: 10.1007/978-1-4757-0602-4_18.

Chaum, D. 1983. "Blind Signature System". In: *Advances in Cryptology. CRYPTO '83*. Plenum Press. 153. DOI: 10.1007/978-1-4684-4730-9_14.

Chen, L., C. Kudla, and K. G. Paterson. 2004. "Concurrent Signatures". In: *Advances in Cryptology. EUROCRYPT '04*. Springer. 287–305. ISBN: 978-3-540-24676-3. DOI: 10.1007/978-3-540-24676-3_18.

Chowdhury, O., M. Pontual, W. H. Winsborough, T. Yu, K. Irwin, and J. Niu. 2012. "Ensuring Authorization Privileges for Cascading User Obligations". In: *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies. SACMAT '12*. ACM. 33–44. ISBN: 978-1-4503-1295-0. DOI: 10.1145/2295136.2295144.

Clark, D. D. 1988. "The Design Philosophy of the DARPA Internet Protocols". In: *Symposium Proceedings on Communications Architectures and Protocols. SIGCOMM '88*. ACM. 106–114. ISBN: 0-89791-279-9. DOI: 10.1145/52324.52336.

Cohen, R. and Y. Lindell. 2017. "Fairness Versus Guaranteed Output Delivery in Secure Multiparty Computation". *J. Cryptol.* 30(4): 1157–1186. ISSN: 0933-2790. DOI: 10.1007/s00145-016-9245-5.

Corin, R., S. Etalle, J. den Hartog, G. Lenzini, and I. Staicu. 2005. "A Logic for Auditing Accountability in Decentralized Systems". In: *Formal Aspects in Security and Trust: IFIP TC1 WG1.7 Workshop on Formal Aspects in Security and Trust. FAST '05*. Springer US. 187–201. ISBN: 978-0-387-24098-5. DOI: 10.1007/0-387-24098-5_14.

Corrigan-Gibbs, H. and B. Ford. 2010. "Dissent: Accountable Anonymous Group Messaging". In: *Proceedings of the 17th Conference on Computer and Communications Security. CCS '10*. ACM. 340–350. DOI: 10.1145/1866307.1866346.

Datta, A. 2014. "Privacy Through Accountability: A Computer Science Perspective". In: *Proceedings of the 10th International Conference on Distributed Computing and Internet Technology. ICDCIT '14*. Springer-Verlag. 43–49. ISBN: 978-3-319-04482-8. DOI: 10.1007/978-3-319-04483-5_5.

Datta, A., D. Garg, D. Kaynar, D. Sharma, and A. Sinha. 2015. "Program Actions As Actual Causes: A Building Block for Accountability". In: *Proceedings of the IEEE 28th Computer Security Foundations Symposium. CSF '15*. IEEE Computer Society. 261–275. ISBN: 978-1-4673-7538-2. DOI: 10.1109/CSF.2015.25.

Diakopoulos, N., S. Friedler, M. Arenas, S. Barocas, M. Hay, B. Howe, H. V. Jagadish, K. Unsworth, A. Sahuguet, S. Venkatasubramanian, C. Wilson, C. Yu, and B. Zevenbergen. "FAT/ML Principles for Accountable Algorithms and a Social Impact Statement for Algorithms". Accessed August 31, 2020. URL: https://www.fatml.org/resources/principles-for-accountable-algorithms.

Dingledine, R., M. J. Freedman, and D. Molnar. 2001. "Accountability". In: *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. Ed. by A. Oram. Sebastopol, CA, USA: O'Reilly & Associates, Inc. Chap. 16. ISBN: 059600110X.

Dong, Y., C. Choi, and Z.-L. Zhang. 2006. "LIPS: A Lightweight Permit System for Packet Source Origin Accountability". *Comput. Netw.* 50(18): 3622–3641. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2006.03.003.

Estrin, D., J. C. Mogul, and G. Tsudik. 1989. "Visa Protocols for Controlling Interorganizational Datagram Flow". *IEEE J. Sel. Areas Commun.* 7(4): 486–498. ISSN: 0733-8716. DOI: 10.1109/49.17712.

Estrin, D. and G. Tsudik. 1987. "*Visa* Scheme for Inter-Organization Network Security". In: *IEEE Symposium on Security and Privacy. SP '87*. IEEE. 174–183. DOI: 10.1109/SP.1987.10002.

EU Article 29 Working Party. 2010. "Opinion 3/2010 on the principle of accountability".

Farkas, C., G. Ziegler, A. Meretei, and A. Lörincz. 2002. "Anonymity and accountability in self-organizing electronic communities". In: *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society. WPES '02*. ACM. 81–90. ISBN: 1-58113-633-1. DOI: 10.1145/644527.644536.

Feigenbaum, J., A. D. Jaggard, and R. N. Wright. 2011. "Towards a formal model of accountability". In: *Proceedings of the 2011 New Security Paradigms Workshop. NSPW '11*. ACM. 45–56. ISBN: 978-1-4503-1078-9. DOI: 10.1145/2073276.2073282.

Feigenbaum, J., A. D. Jaggard, and R. N. Wright. 2014. "Open vs. Closed Systems for Accountability". In: *Proceedings of the 2014 Symposium and Bootcamp on the Science of Security. HotSoS '14*. ACM. 4:1–4:11. ISBN: 978-1-4503-2907-1. DOI: 10.1145/2600176.2600179.

Feigenbaum, J., A. D. Jaggard, R. N. Wright, and H. Xiao. 2012. "Systematizing "Accountability" in Computer Science". *Tech. rep.* No. 1452. Accessed November 12, 2020. Yale University Department of Computer Science. URL: https://www.cs.yale.edu/publications/techreports/tr1452.pdf.

Felici, M., T. Koulouris, and S. Pearson. 2013. "Accountability for Data Governance in Cloud Ecosystems". In: *IEEE 5th International Conference on Cloud Computing Technology and Science*. Vol. 2. *CloudCom '13*. IEEE Computer Society. 327–332. DOI: 10.1109/CloudCom.2013.157.

Frankle, J., S. Park, D. Shaar, S. Goldwasser, and D. Weitzner. 2018. "Practical Accountability of Secret Processes". In: *27th USENIX Security Symposium. USENIX Security '18*. Accessed August 22, 2020. USENIX Association. 657–674. ISBN: 978-1-939133-04-5. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/frankie.

Garg, D., L. Jia, and A. Datta. 2011. "Policy Auditing over Incomplete Logs: Theory, Implementation and Applications". In: *Proceedings of the 18th ACM Conference on Computer and Communications Security. CCS '11*. ACM. 151–162. ISBN: 978-1-4503-0948-6. DOI: 10.1145/2046707.2046726.

Goldreich, O., S. Micali, and A. Wigderson. 1987. "How to Play ANY Mental Game: or, A Completeness Theorem for Protocols with Honest Majority (Extended Abstract)". In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing. STOC '87*. ACM. 218–229. ISBN: 0-89791-221-7. DOI: 10.1145/28395.28420.

Goldwasser, S. and S. Park. 2017. "Public Accountability vs. Secret Laws: Can They Coexist?: A Cryptographic Proposal". In: *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society. WPES '17*. ACM. 99–110. ISBN: 978-1-4503-5175-1. DOI: 10.1145/3139550.3139565.

Gössler, G. and D. Le Métayer. 2015. "A general framework for blaming in component-based systems". *Sci. Comput. Program.* 113: 223–235. ISSN: 0167-6423. DOI: 10.1016/j.scico.2015.06.010.

Granatyr, J., V. Botelho, O. R. Lessing, E. E. Scalabrin, J.-P. Barthès, and F. Enembreck. 2015. "Trust and Reputation Models for Multiagent Systems". *ACM Comput. Surv.* 48(2): 27:1–27:42. ISSN: 0360-0300. DOI: 10.1145/2816826.

Grant, R. and R. Keohane. 2005. "Accountability and Abuses of Power in World Politics". *Am. Polit. Sci. Rev.* 99(1): 29–43. DOI: 10.1017/S000305540505147.

Guts, N., C. Fournet, and F. Zappa Nardelli. 2009. "Reliable Evidence: Auditability by Typing". In: *14^{th} European Symposium on Research in Computer Security. ESORICS '09*. Springer. 168–183. ISBN: 978-3-642-04444-1. DOI: 10.1007/978-3-642-04444-1_11.

Haeberlen, A. 2010. "A Case for the Accountable Cloud". *SIGOPS Oper. Syst. Rev.* 44(2): 52–57. ISSN: 0163-5980. DOI: 10.1145/1773912.1773926.

Haeberlen, A., P. Aditya, R. Rodrigues, and P. Druschel. 2010. "Accountable Virtual Machines". In: *Proceedings of the 9^{th} USENIX Conference on Operating Systems Design and Implementation. OSDI '10*. Accessed November 23, 2020. USENIX Association. 119–134. URL: https://www.usenix.org/legacy/events/osdi10/tech/full_papers/Haeberlen.pdf.

Haeberlen, A., I. Avramopoulos, J. Rexford, and P. Druschel. 2009. "NetReview: Detecting when Interdomain Routing Goes Wrong". In: *Proceedings of the 6^{th} USENIX Symposium on Networked Systems Design and Implementation. NSDI '09*. Accessed November 23, 2020. USENIX Association. 437–452. URL: https://www.usenix.org/legacy/events/nsdi09/tech/full_papers/haeberlen/haeberlen.pdf.

Haeberlen, A., P. Kouznetsov, and P. Druschel. 2007. "PeerReview: practical accountability for distributed systems". *SIGOPS Oper. Syst. Rev.* 41(6): 175–188. ISSN: 0163-5980. DOI: 10.1145/1323293.1294279.

Haeberlen, A., R. Rodrigues, K. Gummadi, and P. Druschel. 2008. "Pretty Good Packet Authentication". In: *Proceedings of the Fourth Conference on Hot Topics in System Dependability. HotDep '08.* Accessed August 22, 2020. USENIX Association. URL: https://www.usenix.org/conference/hotdep-08/pretty-good-packet-authentication.

Halpern, J. Y. 2015. "A Modification of the Halpern–Pearl Definition of Causality". In: *Proceedings of the 24th International Conference on Artificial Intelligence. IJCAI '15.* Accessed August 22, 2020. AAAI Press. 3022–3033. ISBN: 978-1-57735-738-4. URL: https://www.ijcai.org/Proceedings/15/Papers/427.pdf.

Halpern, J. Y. 2016. *Actual Causality.* The MIT Press. ISBN: 978-0-26203-502-6.

Halpern, J. Y. and J. Pearl. 2005a. "Causes and Explanations: A Structural-Model Approach. Part I: Causes". *Brit. J. Philos. Sci.* 56(4): 843–887. DOI: 10.1093/bjps/axi147.

Halpern, J. Y. and J. Pearl. 2005b. "Causes and Explanations: A Structural-Model Approach. Part II: Explanations". *Brit. J. Philos. Sci.* 56(4): 889–911. DOI: 10.1093/bjps/axi148.

Hendrikx, F., K. Bubendorfer, and R. Chard. 2015. "Reputation Systems". *J. Parallel Distrib. Comput.* 75(C): 184–197. ISSN: 0743-7315. DOI: 10.1016/j.jpdc.2014.08.004.

Henry, R. and I. Goldberg. 2011. "Formalizing Anonymous Blacklisting Systems". In: *Proceedings of the 2011 IEEE Symposium on Security and Privacy. SP '11.* IEEE Computer Society. 81–95. ISBN: 978-0-7695-4402-1. DOI: 10.1109/SP.2011.13.

Herlihy, M. and M. Moir. 2016. "Enhancing Accountability and Trust in Distributed Ledgers". arXiv: 1606.07490.

Irwin, K., T. Yu, and W. H. Winsborough. 2006. "On the Modeling and Analysis of Obligations". In: *Proceedings of the 13th ACM Conference on Computer and Communications Security. CCS '06.* ACM. 134–143. ISBN: 1-59593-518-5. DOI: 10.1145/1180405.1180423.

Ishai, Y., R. Ostrovsky, and V. Zikas. 2014. "Secure Multi-Party Computation with Identifiable Abort". In: *Advances in Cryptology. CRYPTO '14.* Springer. 369–386. ISBN: 978-3-662-44381-1. DOI: 10.1007/978-3-662-44381-1_21.

Jagadeesan, R., A. Jeffrey, C. Pitcher, and J. Riely. 2009. "Towards a Theory of Accountability and Audit". In: *Proceedings of the 14th European Conference on Research in Computer Security. ESORICS '09*. Springer-Verlag. 152–167. ISBN: 3-642-04443-3. DOI: 10.1007/978-3-642-04444-1_10.

Jøsang, A., R. Ismail, and C. Boyd. 2007. "A Survey of Trust and Reputation Systems for Online Service Provision". *Decis. Support Syst.* 43(2): 618–644. ISSN: 0167-9236. DOI: 10.1016/j.dss.2005.05.019.

Kacianka, S., K. Beckers, F. Kelbert, and P. Kumari. 2017. "How *Accountability* is Implemented and Understood in Research Tools: A Systematic Mapping Study". In: *Proceedings of the 18th International Conference on Product-Focused Software Process Improvement. PROFES '17*. Springer. 199–218. DOI: 10.1007/978-3-319-69926-4_15.

Kailar, R. 1996. "Accountability in Electronic Commerce Protocols". *IEEE Trans. Softw. Eng.* 22(5): 313–328. ISSN: 0098-5589. DOI: 10.1109/32.502224.

Kessler, V. and H. Neumann. 1998. "A sound logic for analysing electronic commerce protocols". In: *Proceedings of the 5th European Symposium on Research in Computer Security. ESORICS '98*. Springer. 345–360. ISBN: 978-3-540-49784-4. DOI: 10.1007/BFb0055874.

Kiayias, A., H.-S. Zhou, and V. Zikas. 2016. "Fair and Robust Multiparty Computation Using a Global Transaction Ledger". In: *Proceedings, Part II, of the 35th Annual International Conference on Advances in Cryptology. EUROCRYPT '16*. Springer-Verlag. 705–734. ISBN: 978-3-662-49895-8. DOI: 10.1007/978-3-662-49896-5_25.

Klonick, K. 2018. "The New Governors: The People, Rules, and Processes Governing Online Speech". *Harv. L. Rev.* 131(6): 1598–1670. Accessed October 26, 2020. URL: https://ssrn.com/abstract=2937985.

Ko, C., D. A. Frincke, T. Goan Jr., T. Heberlein, K. Levitt, B. Mukherjee, and C. Wee. 1993. "Analysis of an Algorithm for Distributed Recognition and Accountability". In: *Proceedings of the 1st ACM Conference on Computer and Communications Security. CCS '93*. ACM. 154–164. ISBN: 0-89791-629-8. DOI: 10.1145/168588.168608.

Koppell, J. G. 2005. "Pathologies of Accountability: ICANN and the Challenge of "Multiple Accountabilities Disorder"". *Public Adm. Rev.* 65(1): 94–108. ISSN: 1540-6210. DOI: 10.1111/j.1540-6210.2005.00434.x.

Kosba, A., A. Miller, E. Shi, Z. Wen, and C. Papamanthou. 2016. "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts". In: *IEEE Symposium on Security and Privacy. SP '16*. IEEE Computer Society. 839–858. DOI: 10.1109/SP.2016.55.

Kroll, J. A. 2020. "Accountability in Computer Systems". In: *The Oxford Handbook of the Ethics of AI*. Ed. by M. D. Dubber, F. Pasquale, and S. Das. Accessed October 23, 2020. Oxford University Press. ISBN: 9780190067397. URL: https://ssrn.com/abstract=3608468.

Kroll, J. A., J. Huey, S. Barocas, E. W. Felten, J. R. Reidenberg, D. G. Robinson, and H. Yu. 2017. "Accountable Algorithms". *U. Pa. L. Rev.* 165(3): 633–706. Accessed November 23, 2020. URL: https://scholarship.law.upenn.edu/penn_law_review/vol165/iss3/3/.

Kroll, J. A. 2015. "Accountable Algorithms". *PhD thesis*. Princeton University.

Kudo, M. 1998. "Electronic Submission Protocol Based on Temporal Accountability". In: *Proceedings of the 14th Annual Computer Security Applications Conference. ACSAC '98*. IEEE Computer Society. 353–363. ISBN: 0818687894. DOI: 10.1109/CSAC.1998.738656.

Kumaresan, R. and I. Bentov. 2014. "How to Use Bitcoin to Incentivize Correct Computations". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. CCS '14*. ACM. 30–41. ISBN: 978-1-4503-2957-6. DOI: 10.1145/2660267.2660380.

Kungpisdan, S. and Y. Permpoontanalarp. 2002. "Practical Reasoning about Accountability in Electronic Commerce Protocols". In: *Proceedings of the 4th International Conference on Information Security and Cryptology. ICISC '01*. Springer. 268–284. ISBN: 978-3-540-45861-6. DOI: 10.1007/3-540-45861-1_21.

Künnemann, R., I. Esiyok, and M. Backes. 2018a. "Automated Verification of Accountability in Security Protocols". Version 1 of 28 May 2018. arXiv: 1805.10891.

Künnemann, R., D. Garg, and M. Backes. 2018b. "Accountability in Security Protocols". Cryptology ePrint Archive, Report 2018/127. https://eprint.iacr.org/2018/127.

Küsters, R., T. Truderung, and A. Vogt. 2010. "Accountability: definition and relationship to verifiability". In: *Proceedings of the 17th ACM conference on Computer and communications security. CCS '10.* ACM. 526–535. ISBN: 978-1-4503-0245-6. DOI: 10.1145/1866307.1866366.

Lamport, L. 1978. "Time, Clocks, and the Ordering of Events in a Distributed System". *Commun. ACM.* 21(7): 558–565. ISSN: 0001-0782. DOI: 10.1145/359545.359563.

Lampson, B. 2005a. "Notes for presentation entitled "Acountability and Freedom"". Available at http://research.microsoft.com/en-us/um/people/blampson/slides/AccountabilityAndFreedom.ppt. Accessed March 20, 2014.

Lampson, B. 2009. "Privacy and Security: Usable Security: How to Get It". *Commun. ACM.* 52(11): 25–27. ISSN: 0001-0782. DOI: 10.1145/1592761.1592773.

Lampson, B. W. 2004. "Computer Security in the Real World". *Computer.* 37(6): 37–46. ISSN: 0018-9162. DOI: 10.1109/MC.2004.17.

Lampson, B. W. 2005b. "Accountability and Freedom". Slides at http://bwlampson.site/Slides/AccountabilityAndFreedom.ppt, accessed 30 August 2018.

Laskowski, P. and J. Chuang. 2006. "Network Monitors and Contracting Systems: Competition and Innovation". In: *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. SIGCOMM '06.* ACM. 183–194. ISBN: 1-59593-308-5. DOI: 10.1145/1159913.1159935.

Lin, K.-J., J. Zou, and Y. Wang. 2010. "Accountability Computing for e-Society". In: *24th IEEE International Conference on Advanced Information Networking and Applications. AINA '10.* 34–41. DOI: 10.1109/AINA.2010.167.

Lindberg, S. I. 2013. "Mapping accountability: core concept and subtypes". *Int. Rev. Adm. Sci.* 79(2): 202–226.

Lindell, A. Y. 2008. "Legally-Enforceable Fairness in Secure Two-Party Computation". In: *Topics in Cryptology. CT-RSA '08*. Springer. 121–137. ISBN: 978-3-540-79263-5. DOI: 10.1007/978-3-540-79263-5_8.

Liu, J., Y. Xiao, and J. Gao. 2011. "Accountability in smart grids". In: *IEEE Consumer Communications and Networking Conference. CCNC '11*. IEEE. 1166–1170. ISBN: 978-1-4244-8790-5. DOI: 10.1109/CCNC.2011.5766360.

Liu, X., X. Yang, D. Wetherall, and T. Anderson. 2006. "Efficient and Secure Source Authentication with Packet Passports". In: *Proceedings of the 2nd Conference on Steps to Reducing Unwanted Traffic on the Internet. SRUTI '06*. Accessed August 22, 2020. USENIX Association. 7–13. URL: https://www.usenix.org/conference/sruti-06/efficient-and-secure-source-authentication-packet-passports.

Marinovic, S., N. Dulay, and M. Sloman. 2014. "Rumpole: An Introspective Break-Glass Access Control Language". *ACM Trans. Inf. Syst. Secur.* 17(1): 2:1–2:32. ISSN: 1094-9224. DOI: 10.1145/2629502.

Maskin, E. and J. Tirole. 2004. "The Politician and the Judge: Accountability in Government". *Am. Econ. Rev.* 94(4): 1034–1054. DOI: 10.1257/0002828042002606.

Micali, S., K. Ohta, and L. Reyzin. 2001. "Accountable-subgroup multisignatures: extended abstract". In: *Proceedings of the 8th ACM conference on Computer and Communications Security. CCS '01*. ACM. 245–254. ISBN: 1-58113-385-5. DOI: 10.1145/501983.502017.

Michalakis, N., R. Soulé, and R. Grimm. 2007. "Ensuring Content Integrity for Untrusted Peer-to-Peer Content Distribution Networks". In: *4th USENIX Symposium on Networked Systems Design & Implementation. NSDI '07*. Accessed November 23, 2020. USENIX Association. 145–158. URL: https://www.usenix.org/conference/nsdi-07/ensuring-content-integrity-untrusted-peer-peer-content-distribution-networks.

MIT Decentralized Information Group. 2009. "Transparent Accountable Datamining Initiative". Accessed November 12, 2020. URL: http://dig.csail.mit.edu/TAMI/.

MIT Decentralized Information Group. 2010. "Social Web Privacy". Accessed November 12, 2020. URL: http://dig.csail.mit.edu/2009/SocialWebPrivacy/.

MIT Decentralized Information Group. 2011. "Theory and Practice of Accountable Systems". Accessed November 12, 2020. URL: http://dig.csail.mit.edu/2009/NSF-TPAS/index.html.

Mracek, J. 1983. "Network Access Control in Multi-Net Internet Transport". S.B. Thesis, M.I.T. EECS Department.

Mulgan, R. 2000. "'Accountability': An Ever-Expanding Concept?" *Public Adm.* 78(3): 555–573. ISSN: 1467-9299. DOI: 10.1111/1467-9299.00218.

Nakamoto, S. 2008. "Bitcoin: A Peer-to-Peer Electronic Cash System". https://bitcoin.org/bitcoin.pdf, accessed October 16, 2020.

National Research Council. 1991. *Computers at Risk: Safe Computing in the Information Age*. Washington, DC: The National Academies Press. ISBN: 978-0-309-04388-5. DOI: 10.17226/1581.

Naylor, D., M. K. Mukerjee, and P. Steenkiste. 2014. "Balancing Accountability and Privacy in the Network". In: *Proceedings of the 2014 ACM Conference on SIGCOMM. SIGCOMM '14*. ACM. 75–86. ISBN: 978-1-4503-2836-4. DOI: 10.1145/2619239.2626306.

Nissenbaum, H. 1997. "Accountability in a Computerized Society". In: *Human Values and the Design of Computer Technology*. Ed. by B. Friedman. Stanford, CA, USA: Center for the Study of Language and Information. 41–64. ISBN: 1-57586-080-5.

Nuñez, D., C. Fernandez-Gago, S. Pearson, and M. Felici. 2013. "A Metamodel for Measuring Accountability Attributes in the Cloud". In: *IEEE 5th International Conference on Cloud Computing Technology and Science*. Vol. 1. *CloudCom '13*. IEEE Computer Society. 355–362. DOI: 10.1109/CloudCom.2013.53.

Ó Coileáin, D. and D. O'Mahony. 2014a. "Savant: A framework for supporting content accountability in information centric networks". In: *10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness. QSHINE '14*. IEEE. 188–190. DOI: 10.1109/QSHINE.2014.6928686.

Ó Coileáin, D. and D. O'Mahony. 2014b. "SAVANT: Aggregated Feedback and Accountability Framework for Named Data Networking". In: *Proceedings of the 1st ACM Conference on Information-Centric Networking. ACM-ICN '14*. ACM. 187–188. ISBN: 978-1-4503-3206-4. DOI: 10.1145/2660129.2660165.

Ó Coileáin, D. and D. O'Mahony. 2015. "Accounting and Accountability in Content Distribution Architectures: A Survey". *ACM Comput. Surv.* 47(4): 59:1–59:35. ISSN: 0360-0300. DOI: 10.1145/2723701.

Oh, S. E., J. Y. Chun, L. Jia, D. Garg, C. A. Gunter, and A. Datta. 2014. "Privacy-preserving Audit for Broker-based Health Information Exchange". In: *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy. CODASPY '14*. ACM. 313–320. ISBN: 978-1-4503-2278-2. DOI: 10.1145/2557547.2557576.

Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. 1st ed. New York, NY, USA: Cambridge University Press. ISBN: 0-521-77362-8.

Pearson, S. and N. Wainwright. 2013. "An interdisciplinary approach to accountability for future internet service provision". *Int. J. Trust Manage. in Comput. and Commun.* 1(1): 52–72. ISSN: 2048-8386, 2048-8378. DOI: 10.1504/IJTMCC.2013.052524.

Pontual, M., O. Chowdhury, W. H. Winsborough, T. Yu, and K. Irwin. 2011. "On the Management of User Obligations". In: *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies. SACMAT '11*. ACM. 175–184. ISBN: 978-1-4503-0688-1. DOI: 10.1145/1998441.1998473.

Raab, C. 2012. "The Meaning of "Accountability" in the Information Privacy Context". In: *Managing Privacy through Accountability*. Ed. by D. Guagnin, L. Hempel, C. Ilten, I. Kroener, D. Neyland, and H. Postigo. 1st ed. Palgrave Macmillan. 15–32. ISBN: 978-0 230 36932-0. DOI: 10.1057/9781137032225.

Radin, B. A. and B. S. Romzek. 1996. "Accountability Expectations in an Intergovernmental Arena: The National Rural Development Partnership". *Publius*. 26(2): 59–81. ISSN: 0048-5950. DOI: 10.1093/oxfordjournals.pubjof.a029855.

Resnick, P., R. Zeckhauser, E. Friedman, and K. Kuwabara. 2000. "Reputation Systems". *Commun. ACM*. 43(12): 45–48. ISSN: 0001-0782. DOI: 10.1145/355112.355122.

Rivest, R. L. and A. Shamir. 1997. "PayWord and MicroMint: Two Simple Micropayment Schemes". In: *Proceedings of the International Workshop on Security Protocols. Security Protocols '96*. Springer-Verlag. 69–87. ISBN: 3-540-62494-5. DOI: 10.1007/3-540-62494-5_6.

Ruffing, T., A. Kate, and D. Schröder. 2015. "Liar, Liar, Coins on Fire!: Penalizing Equivocation By Loss of Bitcoins". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. CCS '15*. ACM. 219–230. ISBN: 978-1-4503-3832-5. DOI: 10.1145/2810103.2813686.

Sandhu, R. and P. Samarati. 1996. "Authentication, Access Control, and Audit". *ACM Comput. Surv.* 28(1): 241–243. ISSN: 0360-0300. DOI: 10.1145/234313.234412.

Schedler, A., L. Diamond, and M. F. Plattner, eds. 1999. *The Self-Restraining State: Power and Accountability in New Democracies*. Lynne Rienner Publishers, Inc.

Sharma, D. 2015. "Interaction-aware Actual Causation: A Building Block for Accountability in Security Protocols". *PhD thesis*. Carnegie Mellon University.

Sullivan, K., J. Clarke, and B. P. Mulcahy. 2010. "Trust-terms Ontology for Defining Security Requirements and Metrics". In: *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. ECSA '10*. ACM. 175–180. ISBN: 978-1-4503-0179-4. DOI: 10.1145/1842752.1842789.

Syta, E., H. Corrigan-Gibbs, S.-C. Weng, D. Wolinsky, B. Ford, and A. Johnson. 2014. "Security Analysis of Accountable Anonymity in Dissent". *ACM Trans. Inf. Syst. Secur.* 17(1): 4:1–4:35. ISSN: 1094-9224. DOI: 10.1145/2629621.

Syverson, P. and G. Boyce. 2016. "Bake in .Onion for Tear-Free and Stronger Website Authentication". *IEEE Security and Privacy*. 14(2): 15–21. ISSN: 1540-7993. DOI: 10.1109/MSP.2016.33.

Tsang, P. P., M. H. Au, A. Kapadia, and S. W. Smith. 2008. "PEREA: towards practical TTP-free revocation in anonymous authentication". In: *Proceedings of the 15th ACM conference on Computer and communications security. CCS '08*. ACM. 333–344. ISBN: 978-1-59593-810-7. DOI: 10.1145/1455770.1455813.

Tsang, P. P., A. Kapadia, C. Cornelius, and S. W. Smith. 2011. "Nymble: Blocking Misbehaving Users in Anonymizing Networks". *IEEE Trans. Dependable Secur. Comput.* 8(2): 256–269. ISSN: 1545-5971. DOI: 10.1109/TDSC.2009.38.

Uhler, O. M., H. Coursier, F. Siordet, C. Pilloud, R. Boppe, R.-J. Wilhelm, and J.-P. Schoenholzer. 1958. *Commentary on The Geneva Conventions of 12 August 1949: IV Geneva Convention relative to the protection of civilian persons in time of war*. Ed. by J. S. Pictet. Translated by Ronald Griffin and C. W. Dumbleton. Accessed 10 May 2018. Geneva: International Committee of the Red Cross. URL: https://www.loc.gov/rr/frd/Military_Law/pdf/GC_1949-IV.pdf.

Vaughan, J. A., L. Jia, K. Mazurak, and S. Zdancewic. 2008. "Evidence-Based Audit". In: *Proceedings of the 21st IEEE Computer Security Foundations Symposium. CSF '08*. IEEE Computer Society. 177–191. ISBN: 978-0-7695-3182-3. DOI: 10.1109/CSF.2008.24.

Weisband, E. and A. Ebrahim. 2007. "Introduction: Forging Global Accountabilities". In: *Forging Global Accountabilities: Participation, Pluralism, and Public Ethics*. Ed. by A. Ebrahim and E. Weisband. Cambridge University Press.

Weitzner, D. J. 2017. Private communication.

Weitzner, D. J., H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G. J. Sussman. 2008. "Information Accountability". *Commun. ACM*. 51(6): 82–87. ISSN: 0001-0782. DOI: 10.1145/1349026.1349043.

Wieringa, M. 2020. "What to Account for When Accounting for Algorithms: A Systematic Literature Review on Algorithmic Accountability". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. FAT* '20*. ACM. 1–18. ISBN: 9781450369367. DOI: 10.1145/3351095.3372833.

Wolinsky, D. I., H. Corrigan-Gibbs, B. Ford, and A. Johnson. 2012. "Dissent in Numbers: Making Strong Anonymity Scale". In: *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation. OSDI '12*. Accessed August 22, 2020. USENIX Association. 179–192. ISBN: 978-1-931971-96-6. URL: https://www.usenix.org/conference/osdi12/technical-sessions/presentation/wolinsky.

Yumerefendi, A. R. and J. S. Chase. 2004. "Trust but verify: accountability for network services". In: *Proceedings of the 11th ACM SIGOPS European Workshop. EW '04*. ACM. DOI: 10.1145/1133572.1133585.

Yumerefendi, A. R. and J. S. Chase. 2005. "The role of accountability in dependable distributed systems". In: *Proceedings of the First Workshop on Hot Topics in System Dependability. HotDep '05*. Accessed November 23, 2020. USENIX Association. URL: http://www.hotdep.org/2005.

Yumerefendi, A. R. and J. S. Chase. 2007. "Strong accountability for network storage". *ACM Trans. Storage*. 3(3). ISSN: 1553-3077. DOI: 10.1145/1288783.1288786.

Zhou, J. and D. Gollman. 1996. "A fair non-repudiation protocol". In: *Proceedings of the 1996 IEEE Symposium on Security and Privacy. SP '96*. IEEE Computer Society. 55–61. ISBN: 0-8186-7417-2. DOI: 10.1109/SECPRI.1996.502669.

Zou, J., Y. Wang, and K.-J. Lin. 2010. "A Formal Service Contract Model for Accountable SaaS and Cloud Services". In: *IEEE Seventh International Conference on Services Computing. SCC '10*. IEEE Computer Society. 73–80. DOI: 10.1109/SCC.2010.85.

Index