Practical Accountability of Secret Processes

Jonathan Frankle Sunoo Park Daniel Shaar Shafi Goldwasser Daniel J. Weitzner

Massachusetts Institute of Technology

Abstract

The US federal court system is exploring ways to improve the accountability of electronic surveillance, an opaque process often involving cases *sealed* from public view and tech companies subject to *gag orders* against informing surveilled users. One judge has proposed publicly releasing some metadata about each case on a paper *cover sheet* as a way to balance the competing goals of (1) secrecy, so the target of an investigation does not discover and sabotage it, and (2) accountability, to assure the public that surveillance powers are not misused or abused.

Inspired by the courts' accountability challenge, we illustrate how accountability and secrecy are simultaneously achievable when modern cryptography is brought to bear. Our system improves configurability while preserving secrecy, offering new tradeoffs potentially more palatable to the risk-averse court system. Judges, law enforcement, and companies publish commitments to surveillance actions, argue in zero-knowledge that their behavior is consistent, and compute aggregate surveillance statistics by multi-party computation (MPC).

We demonstrate that these primitives perform efficiently at the scale of the federal judiciary. To do so, we implement a hierarchical form of MPC that mirrors the hierarchy of the court system. We also develop statements in succinct zero-knowledge (SNARKs) whose specificity can be tuned to calibrate the amount of information released. All told, our proposal not only offers the court system a flexible range of options for enhancing accountability in the face of necessary secrecy, but also yields a general framework for accountability in a broader class of *secret information processes*.

1 Introduction

We explore the challenge of providing public accountability for secret processes. To do so, we design a system that increases transparency and accountability for one of the leading United States electronic surveillance laws, the Electronic Communications Privacy Act (ECPA) [2], which allows law enforcement agencies to request data about users from tech companies. The core accountability challenge in the operation of ECPA is that many of the official acts of the judges, law enforcement agencies, and companies remain hidden from public view (*sealed*), often indefinitely. Therefore, the public has limited information on which to base confidence in the system.

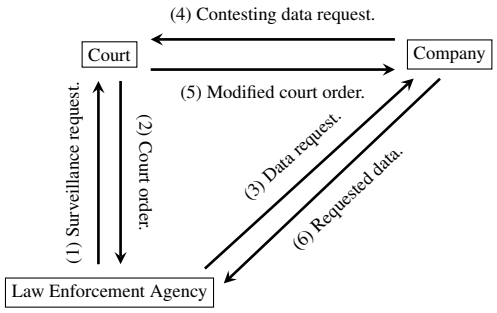


Figure 1: The workflow of electronic surveillance.

To put this in perspective: in 2016, Google received 27,850 requests from US law enforcement agencies for data implicating 57,392 user accounts [4], and Microsoft received 9,907 requests implicating 24,288 users [7]. These numbers, taken from the companies' own voluntary transparency reports, are some of the only publicly available figures on the scope of law enforcement requests for data from technology companies under ECPA.

Underlying many of these requests is a court order. A court order is an action by a *federal judge* requiring a *company* to turn over data related to a *target* (i.e., a user) who is suspected of committing a crime; it is issued in response to a request from a *law enforcement agency*. ECPA is one of several electronic surveillance laws, and each follows somewhat different legal procedures; however, they broadly tend to follow the idealized workflow

in Figure 1. First, a law enforcement agency presents a surveillance request to a federal judge (arrow 1).The judge can either approve or deny it. Should the judge approve the request, she signs an order authorizing the surveillance (arrow 2). A law enforcement agency then presents this order, describing the data to be turned over, to a company (arrow 3). The company either complies or contests the legal basis for the order with the judge (arrow 4). Should the company's challenge be accepted, the order could be narrowed (arrow 5) or eliminated; if not, the company turns over the requested data (arrow 6).

These court orders are the primary procedural marker that surveillance ever took place. They are often *sealed*, i.e., temporarily hidden from the public for a period of time after they are issued. In addition, companies are frequently *gagged*, i.e., banned from discussing the order with the target of the surveillance. These measures are vital for the investigative process: were a target to discover that she were being surveilled, she could change her behavior, endangering the underlying investigation.

According to Judge Stephen Smith, a federal magistrate judge whose role includes adjudicating requests for surveillance, gags and seals come at a cost. Openness of judicial proceedings has long been part of the common-law legal tradition, and court documents are presumed to be public by default. To Judge Smith, a court's public records are "the source of its own legitimacy" [37]. Judge Smith has noted several specific ways that gags and seals undermine the legal mechanisms meant to balance the powers of investigators and those investigated [37]:

1. Indefinite sealing. Many sealed orders are ultimately forgotten by the courts which issued them, meaning ostensibly temporary seals become permanent in practice. To determine whether she was surveilled, a member of the public would have to somehow discover the existence of a sealed record, confirm the seal had expired, and request the record. Making matters worse, these records are scattered across innumerable courthouses nationwide.

2. Inadequate incentive and opportunity to appeal. Seals and gags make it impossible for a target to learn she is being surveilled, let alone contest or appeal the decision. Meanwhile no other party has the incentive to appeal. Companies prefer to reduce compliance and legal costs by cooperating. A law enforcement agency would only consider appealing when a judge denies its request; however, Judge Smith explains that even then, agencies often prefer not to "risk an appeal that could make 'bad law'" by creating precedent that makes surveillance harder in the future. As a result, judges who issue these orders have "literally no appellate guidance."

3. Inability to discern the extent of surveillance. Judge Smith laments that lack of data means "neither Congress nor the public can accurately assess the breadth and depth of current electronic surveillance activity" [38]. Several small efforts shed some light on this process: wiretap reports by the Administrative Office of the US Courts [9] and the aforementioned "transparency reports" by tech companies [7, 4]. These reports, while valuable, clarify only the faintest outlines of surveillance.

The net effect is that electronic surveillance laws are not subject to the usual process of challenge, critique, and modification that keeps the legal system operating within the bounds of constitutional principles. This lack of scrutiny ultimately reduces public trust: we lack answers to many basic questions. Does surveillance abide by legal and administrative rules? Do agencies present authorized requests to companies, and do companies return the minimum amount of data to comply? To a public concerned about the extent of surveillance, credible assurances would increase trust. To foreign governments that regulate cross-border dataflows, such assurances could determine whether companies have to drastically alter data management when operating abroad. Yet, today, no infrastructure for making such assurances exists.

To remedy these concerns, Judge Smith proposes that each order be accompanied by a publicly available *cover sheet* containing general metadata about an order (e.g., kind of data searched, crimes suspected, length of the seal, reasons for sealing) [38]. The cover sheet would serve as a visible marker of sealed cases; when a seal expires, the public can hold the court accountable by requesting the sealed document. Moreover, the cover sheet metadata enables the public to compute aggregate statistics about surveillance, complementing the transparency reports released by the government and companies.

Designing the cover sheet involves balancing two competing instincts: (1) for law enforcement to conduct effective investigations, some information about surveillance must be hidden and (2) public scrutiny can hold law enforcement accountable and prevent abuses of power. The primary design choice available is the amount of information to release.

Our contribution. As a simple sheet of paper, Judge Smith's proposal is inherently limited in its ability to promote public trust while maintaining secrecy. Inspired by Judge Smith's proposal, we demonstrate the accountability achievable when the power of modern cryptography is brought to bear. Cryptographic commitments can indicate the existence of a surveillance document without revealing its contents. Secure multiparty computation (MPC) can allow judges to compute aggregate statistics about all cases—information currently scattered across voluntary transparency reports—without revealing data about any particular case. Zero-knowledge arguments

can demonstrate that a particular surveillance action (e.g., requesting data from a company) follows properly from a previous surveillance action (e.g., a judge's order) without revealing the contents of either item. All of this information is stored on an append-only ledger, giving the courts a way to release information and the public a definitive place to find it. Courts can post additional information to the ledger, from the date that a seal expires to the entirety of a cover sheet. Together, these primitives facilitate a flexible accountability strategy that can provide greater assurance to the public while protecting the secrecy of the investigative process.

To show the practicality of these techniques, we evaluate MPC and zero-knowledge protocols that amply scale to the size of the federal judiciary.[1] To meet our efficiency requirements, we design a hierarchical MPC protocol that mirrors the structure of the federal court system. Our implementation supports sophisticated aggregate statistics (e.g., "how many judges ordered data from Google more than ten times?"), and scales to hundreds of judges who may not stay online long enough to participate in a synchronized multiround protocol. We also implement succinct zero-knowledge arguments about the consistency of data held in different commitments; the legal system can tune the specificity of these statements in order to calibrate the amount of information released. Our implementations apply and extend the existing libraries Webmpc [16, 29] and Jiff [5] (for MPC) and libSNARK [34] (for zero-knowledge). Our design is not coupled to these specific libraries, however; an analogous implementation could be developed based on any suitable MPC and SNARK libraries. Thus, our design can straightforwardly inherit efficiency improvements of future MPC and SNARK libraries.

Finally, we observe that the federal court system's accountability challenge is an instance of a broader class of *secret information processes*, where some information must be kept secret among participants (e.g., judges, law enforcement agencies, and companies) engaging in a protocol (e.g., surveillance as in Figure 1), yet the propriety of the participants' interactions are of interest to an auditor (e.g., the public). After presenting our system as tailored to the case study of electronic surveillance, we describe a framework that generalizes our strategy to any accountability problem that can be framed as a secret information process. Concrete examples include clinical trials, public spending, and other surveillance regimes.

In summary, we design a novel system achieving public accountability for secret processes while leveraging off-the-shelf cryptographic primitives and libraries. The design is adaptable to new legal requirements, new transparency goals, and entirely new applications within the realm of secret information processes.

[1] There are approximately 900 federal judges [10].

Roadmap. Section 2 discusses related work. Section 3 introduces our threat model and security goals. Section 4 introduces the system design of our accountability scheme for the court system, and Section 5 presents detailed protocol algorithms. Sections 6 and 7 discuss the implementation and performance of hierarchical MPC and succinct zero knowledge. Section 8 generalizes our framework to a range of scenarios beyond electronic surveillance, and Section 9 concludes.

2 Related Work

Accountability. The term *accountability* has many definitions. [21] categorizes technical definitions of accountability according to the timing of interventions, information used to assess actions, and response to violations; [20] further formalizes these ideas. [31] surveys definitions from both computer science and law. [44] surveys definitions specific to distributed systems and the cloud.

In the terminology of these surveys, our focus is on *detection* ("The system facilitates detection of a violation" [21]) and *responsibility* ("Did the organization follow the rules?" [31]). Our additional challenge is that we consider protocols that occur in secret. Other accountability definitions consider how "violations [are] tied to punishment" [21, 28]; we defer this question to the legal system and consider it beyond the scope of this work. Unlike [32], which advocates for "prospective" accountability measures like access control, our view of accountability is entirely retrospective.

Implementations of accountability in settings where remote computers handle data (e.g., the cloud [32, 39, 40] and healthcare [30]) typically follow the transparency-centric blueprint of *information accountability* [43]: remote actors record their actions and make logs available for scrutiny by an auditor (e.g, a user). In our setting (electronic surveillance), we strive to release as little information as possible subject to accountability goals, meaning complete transparency is not a solution.

Cryptography and government surveillance. Kroll, Felten, and Boneh [27] also consider electronic surveillance but focus on cryptographically ensuring that participants only have access to data when legally authorized. Such access control is orthogonal to our work. Their system includes an audit log that records all surveillance actions; much of their logged data is encrypted with a "secret escrow key." In contrast, motivated by concerns articulated directly by the legal community, we focus exclusively on accountability and develop a nuanced framework for *public* release of controlled amounts of information to address a general class of accountability problems, of which electronic surveillance is one instance.

Bates et al. [12] consider adding accountability to court-sanctioned wiretaps, in which law enforcement agencies can request phone call content. They encrypt duplicates of all wiretapped data in a fashion only accessible by courts and other auditors and keep logs thereof such that they can later be analyzed for aggregate statistics or compared with law enforcement records. A key difference between [12] and our system is that our design enables the public to directly verify the propriety of surveillance activities, partially in real time.

Goldwasser and Park [23] focus on a different legal application: secret laws in the context of the Foreign Intelligence Surveillance Act (FISA) [3], where the operations of the court applying the law is secret. Succinct zero-knowledge is used to certify consistency of recorded actions with unknown judicial actions. While our work and [23] are similar in motivation and share some cryptographic tools, Goldwasser and Park address a different application. Moreover, our paper differs in its implementations demonstrating practicality and its consideration of aggregate statistics. Unlike this work, [23] does not model parties in the role of companies.

Other research that suggests applying cryptography to enforce rules governing access-control aspects of surveillance includes: [25], which enforces privacy for NSA telephony metadata surveillance; [36], which uses private set intersection for surveillance involving joins over large databases; and [35], which uses the same technique for searching communication graphs.

Efficient MPC and SNARKs. LibSNARK [34] is the primary existing implementation of SNARKs. (Other libraries are in active development [1, 6].) More numerous implementation efforts have been made for MPC under a range of assumptions and adversary models, e.g., [16, 29, 5, 11, 42, 19]. The idea of placing most of the workload of MPC on a subset of parties has been explored before, (e.g., constant-round protocols by [18, 24]); we build upon this literature by designing a hierarchically structured MPC protocol specifically to match the hierarchy of the existing US court system.

3 Threat Model and Security Goals

Our high-level policy goals are to hold the electronic surveillance process accountable to the public by (1) demonstrating that each participant performs its role properly and stays within the bounds of the law and (2) ensuring that the public is aware of the general extent of government surveillance. The accountability measures we propose place checks on the behavior of judges, law enforcement agencies, and companies. Such checks are important against oversight as well as malice, as these participants can misbehave in a number of ways. For example, as Judge Smith explains, forgetful judges may lose track of orders whose seals have expired. More maliciously, in 2016, a Brooklyn prosecutor was arrested for "spy[ing] on [a] love interest" and "forg[ing] judges' signatures to keep the eavesdropping scheme running for about a year" [22].

Our goal is to achieve public accountability even in the face of unreliable and untrustworthy participants. Next, we specify our *threat model* for each type of participant in the system, and enumerate the *security goals* that, if met, will make it possible to maintain accountability under this threat model.

3.1 Threat model

Our threat model considers the three parties presented in Figure 1—judges, law enforcement agencies, and companies—along with the public. Their roles and the assumptions we make about each are described below. We assume all parties are computationally bounded.

Judges. Judges consider requests for surveillance and issue court orders that allow law enforcement agencies to request data from companies. We must consider judges in the context of the courts in which they operate, which include staff members and possibly other judges. We consider courts to be honest-but-curious: they will adhere to the designated protocols, but should not be able to learn internal information about the workings of other courts. Although one might argue that the judges themselves can be trusted with this information, we do not trust their staffs. Hereon, we use the terms "judge" and "court" interchangeably to refer to an entire courthouse.

In addition, when it comes to sealed orders, judges may be forgetful: as Judge Smith observes, judges frequently fail to unseal orders when the seals have expired [38].

Law enforcement agencies. Law enforcement agencies make requests for surveillance to judges in the context of ongoing investigations. If these requests are approved and a judge issues a court order, a law enforcement agency may request data from the relevant companies. We model law enforcement agencies as malicious: e.g., they may forge or alter court orders in order to gain access to unauthorized information (as in the case of the Brooklyn prosecutor [22]).

Companies. Companies possess the data that law enforcement agencies may request if they hold a court order. Companies may optionally contest these orders and, if the order is upheld, must supply the relevant data to the law enforcement agency. We model companies as

malicious: e.g., they might wish to contribute to unauthorized surveillance while maintaining the outside appearance that they are not. Specifically, although companies currently release aggregate statistics about their involvement in the surveillance process [4, 7], our system does not rely on their honesty in reporting these numbers. Other malicious behavior might include colluding with law enforcement to release more data than a court order allows or furnishing data in the absence of a court order.

The public. We model the public as malicious, as the public may include criminals who wish to learn as much as possible about the surveillance process in order to avoid being caught.[2]

Remark 3.1. *Our system requires the parties involved in surveillance to post information to a shared ledger at various points in the surveillance process. Correspondence between logged and real-world events is an aspect of any log-based record-keeping scheme that cannot be enforced using technological means alone. Our system is designed to encourage parties to log honestly or report dishonest logging they observe (see Remark 4.1). Our analysis focuses on the cryptographic guarantees provided by the system, however, rather than a rigorous game-theoretic analysis of incentive-based behavior. Most of this paper therefore assumes that surveillance orders and other logged events are recorded correctly, except where otherwise noted.*

3.2 Security Goals

In order to achieve accountability in light of this threat model, our system will need to satisfy three high-level security goals.

Accountability to the public. The system must reveal enough information 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.

Correctness. All of the information that our system computes and reveals must be correct. The aggregate statistics it computes and releases to the public must accurately reflect the state of electronic surveillance. Any assurances that our system makes to the public about the (im)propriety of the electronic surveillance process must be reported accurately.

Confidentiality. The public must not learn information that could undermine the investigative process. None of the other parties (courts, law enforcement agencies, and companies) may learn any information beyond that which they already know in the current ECPA process and that which is released to the public.

For particularly sensitive applications, the confidentiality guarantee should be *perfect (information-theoretic)*: this means confidentiality should hold unconditionally, even against arbitrarily powerful adversaries that may be computationally unbounded.[3] A *perfect confidentiality* guarantee would be of particular importance in contexts where unauthorized breaks of confidentiality could have catastrophic consequences (such as national security). We envision that a truly unconditional confidentiality guarantee could catalyze the consideration of accountability systems in contexts involving very sensitive information where decision-makers are traditionally risk-averse, such as the court system.

4 System Design

We present the design of our proposed system for accountability in electronic surveillance. Section 4.1 informally introduces four cryptographic primitives and their security guarantees.[4] Section 4.2 outlines the configuration of the system—where data is stored and processed. Section 4.3 describes the workflow of the system in relation to the surveillance process summarized in Figure 1. Section 4.4 discusses the packages of design choices available to the court system, exploiting the flexibility of the cryptographic tools to offer a range of options that trade off between secrecy and accountability.

4.1 Cryptographic Tools

Append-only ledgers. An *append-only ledger* is a log containing an ordered sequence of data consistently visi-

[2] By placing all data on an immutable public ledger and giving the public no role in our system besides that of observer, we effectively reduce the public to a passive adversary.

[3] This is in contrast to *computational* confidentiality guarantees, which provide confidentiality only against adversaries that are *efficient* or *computationally bounded*. Even with the latter weaker type of guarantee, it is possible to ensure confidentiality against any adversary with computing power within the realistically foreseeable future; *computational* guarantees are quite common in practice and widely considered acceptable for many applications. One reason to opt for computational guarantees over information-theoretic ones is that typically, information-theoretic guarantees carry some loss in efficiency; however, this benefit may be outweighed in particularly sensitive applications, or when confidentiality is desirable for a very long-term future where advances in computing power are not foreseeable.

[4] For rigorous formal definitions of these cryptographic primitives, we refer to any standard cryptography textbook (e.g., [26]).

ble to anyone (within a designated system), and to which data may be appended over time, but whose contents may not be edited or deleted. The append-only nature of the ledger is key for the maintenance of a *globally consistent* and *tamper-proof* data record over time.

In our system, the ledger records credibly time-stamped information about surveillance events. Typically, data stored on the ledger will cryptographically hide some sensitive information about a surveillance event, while revealing select other information about it for the sake of accountability. Placing information on the ledger is one means by which we reveal information to the public, facilitating the security goal of accountability from Section 3.

Cryptographic commitments. A cryptographic *commitment* c is a string generated from some input data D, which has the properties of *hiding* and *binding*: i.e., c reveals no information about the value of D, and yet D can be revealed or "opened" (by the person who created the commitment) in such a way that any observer can be sure that D is the data with respect to which the commitment was made. We refer to D as the *content* of c.

In our system, commitments indicate that a piece of information (e.g., a court order) exists and that its content can credibly be opened at a later time. Posting commitments to the ledger also establishes the existence of a piece of information *at a given point in time*. Returning to the security goals from Section 3, commitments make it possible to reveal a limited amount of information early on (achieving a degree of accountability) without compromising investigative secrecy (achieving confidentiality). Later, when confidentiality is no longer necessary and information can be revealed (i.e., a seal on an order expires), then the commitment can be opened by its creator to achieve full accountability.

Commitments can be *perfectly (information-theoretically) hiding*, achieving the perfect confidentiality goal of in Section 3.2. A well-known commitment scheme that is perfectly hiding is the Pedersen commitment.[5]

Zero-knowledge. A zero-knowledge argument[6] allows a *prover* P to convince a *verifier* V of a fact without revealing any additional information about the fact in the process of doing so. P can provide to V a tuple (R, x, π) consisting of a *binary relation* R, an *input* x, and a *proof* π, such that the verifier is convinced that $\exists w$ s.t. $(x, w) \in R$ yet cannot infer anything about the witness w. Three properties are required of zero-knowledge arguments: *completeness*, that any true statement can be proven by the honest algorithm P such that V accepts the proof; *soundness*, that no purported proof of a false statement (produced by *any* algorithm P^*) should be accepted by the honest verifier V; and *zero-knowledge*, that the proof π reveals no information beyond what can be inferred just from the desired statement that $(x, w) \in R$.

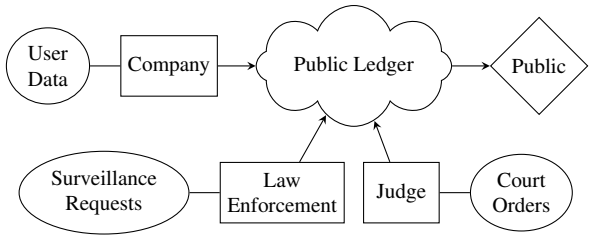


Figure 2: System configuration. Participants (rectangles) read and write to a public ledger (cloud) and local storage (ovals). The public (diamond) reads from the ledger.

In our system, zero-knowledge makes it possible to reveal how secret information relates to a system of rules or to other pieces of secret information without revealing any further information. Concretely our implementation (detailed in Section 7) allows law enforcement to attest (1) knowledge of the content of a commitment c (e.g., to an email address in a request for data made by a law enforcement agency) demonstrating the ability to later *open* c; and (2) that the content of a commitment c is equal to the content of a prior commitment c' (e.g., to an email address in a court order issued by a judge). In case even (2) reveals too much information, our implementation supports not specifying c' exactly, and instead attesting that c' lies in a given set S (e.g., S could include all judges' surveillance authorizations from the last month).

In the terms of our security goals from Section 3, zero knowledge arguments can demonstrate to the public that commitments can be opened and that proper relationships between committed information is preserved (accountability) without revealing any further information about the surveillance process (confidentiality). If these arguments fail, the public can detect when a participant has deviated from the process (accountability).

The SNARK construction [15] that we suggest for use in our system achieves *perfect (information-theoretic) confidentiality*, a goal stated in Section 3.2.[7]

[5] While the Pedersen commitment is not succinct, we note that by combining succinct commitments with perfectly hiding commitments (as also suggested by [23]), it is possible to obtain a commitment that is both succinct and perfectly hiding.

[6] *Zero-knowledge proof* is a more commonly used term than *zero-knowledge argument*. The two terms denote very similar concepts; the difference is lies only in the nature of the *soundness* guarantee (i.e., that false statements cannot be convincingly attested to), which is computational for arguments and statistical for proofs.

[7] In fact, [15] states their secrecy guarantee in a computational (not information-theoretic) form, but their unmodified construction does achieve perfect secrecy and the proofs of [15] suffice unchanged to prove the stronger definition [41]. That perfect zero-knowledge can be achieved is also remarked in the appendix of [14].

Secure multiparty computation (MPC). MPC allows a set of n parties $p_1, \ldots, p_n$, each in possession of private data $x_1, \ldots, x_n$, to jointly compute the output of a function $y = f(x_1, \ldots, x_n)$ on their private inputs. y is computed via an interactive protocol executed by the parties.

Secure MPC makes two guarantees: *correctness* and *secrecy*. Correctness means that the output y is equal to $f(x_1, \ldots, x_n)$. Secrecy means that any adversary that corrupts some subset $S \subset \{p_1, \ldots, p_n\}$ of the parties learns nothing about $\{x_i : p_i \notin S\}$ beyond what can already be inferred given the adversarial inputs $\{x_i : p_i \in S\}$ and the output y. Secrecy is formalized by stipulating that a *simulator* that is given only $(\{x_i : p_i \in S\}, y)$ as input must be able to produce a "simulated" protocol transcript that is indistinguishable from the actual protocol execution run with all the real inputs $(x_1, \ldots, x_n)$.

In our system, MPC enables computation of aggregate statistics about the extent of surveillance across the entire court system through a computation among individual judges. MPC eliminates the need to pool the sensitive data of individual judges in the clear or to defer to companies to compute and release this information piecemeal. In the terms of our security goals, MPC reveals information to the public (accountability) from a source we trust to follow the protocol honestly (the courts) without revealing the internal workings of courts to one another (confidentiality). It also eliminates the need to rely on potentially malicious companies to reveal this information themselves (correctness).

Secret sharing. Secret sharing facilitates our hierarchical MPC protocol. A *secret sharing* of some input data D consists of a set of strings $(D_1, \ldots, D_N)$, called *shares*, satisfying two properties: (1) any subset of $N-1$ shares reveals no information about D, and (2) given all the N shares, D can easily be reconstructed.[8]

Summary. In summary, these cryptographic tools support three high-level properties that we utilize to achieve our security goals:

1. *Trusted records of events:* The append-only ledger and cryptographic commitments create a trustworthy record of surveillance events without revealing sensitive information to the public.
2. *Demonstration of compliance:* Zero-knowledge arguments allow parties to provably assure the public that relevant rules have been followed without revealing any secret information.
3. *Transparency without handling secrets:* MPC enables the court system to accurately compute and release aggregate statistics about surveillance events without ever sharing the sensitive information of individual parties.

[8] For simplicity, we have described so-called "N-out-of-N" secret-sharing. More generally, secret sharing can guarantee that any subset of $k \leq N$ shares enable reconstruction, while any subset of at most $k-1$ shares reveals nothing about D.

4.2 System Configuration

Our system is centered around a publicly visible, append-only ledger where the various entities involved in the electronic surveillance process can post information. As depicted in Figure 2, every judge, law enforcement agency, and company contributes data to this ledger. Judges post cryptographic commitments to all orders issued. Law enforcement agencies post commitments to their activities (warrant requests to judges and data requests to companies), and zero-knowledge arguments about the requests they issue. Companies do the same for the data they deliver to agencies. Members of the public can view and verify all data posted to the ledger.

Each judge, law enforcement agency, and company will need to maintain a small amount of infrastructure: a computer terminal through which to compose posts and local storage (the ovals in Figure 2) to store sensitive information (e.g., the content of sealed court orders). To attest to the authenticity of posts to the ledger, each participant will need to maintain a private signing key and publicize a corresponding verification key. We assume that public-key infrastructure could be established by a reputable party like the Administrative Office of the US Courts.

The ledger itself could be maintained as a distributed system among the participants in the process, a distributed system among a more exclusive group of participants with higher trustworthiness (e.g., the circuit courts of appeals), or by a single entity (e.g., the Administrative Office of the US Courts or the Supreme Court).

4.3 Workflow

Posting to the ledger. The workflow of our system augments the electronic surveillance workflow in Figure 1 with additional information posted to the ledger as depicted in Figure 3. When a judge issues an order (step 2 of Figure 1), she also posts a commitment to the order and additional metadata about the case. At a minimum, this metadata must include the date that the order's seal expires; depending on the system configuration, she could post other metadata (e.g., Judge Smith's cover sheet). The commitment allows the public to later verify that the order was properly unsealed but reveals no information about the commitment's content in the meantime, achieving a degree of accountability in the short-term (while confidentiality is necessary) and full accountability in the long-term (when the seal expires and confidentiality is unnecessary). Since judges are

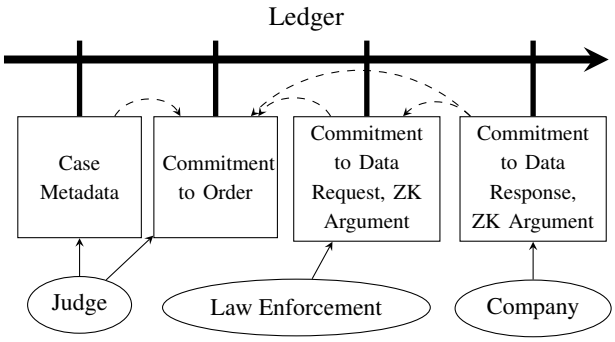


Figure 3: Data posted to the public ledger as the protocol runs. Time moves from left to right. Each rectangle is a post to the ledger. Dashed arrows between rectangles indicate that the source of the arrow could contain a visible reference to the destination. The ovals contain the entities that make each post.

honest-but-curious, they will adhere to this protocol and reliably post commitments whenever new court orders are issued.

The agency then uses this order to request data from a company (step 3 in Figure 1) and posts a commitment to this request alongside a zero-knowledge argument that the request is compatible with a court order (and possibly also with other legal requirements). This commitment, which may never be opened, provides a small amount of accountability within the confines of confidentiality, revealing that *some* law enforcement action took place. The zero-knowledge argument takes accountability a step further: it demonstrates to the public that the law enforcement action was compatible with the original court order (which we trust to have been committed properly), forcing the potentially-malicious law enforcement agency to adhere to the protocol or make public its noncompliance. (Failure to adhere would result in a publicly visible invalid zero-knowledge argument.) If the company responds with matching data (step 6 in Figure 1), it posts a commitment to its response and an argument that it furnished (only) the data implicated by the order and data request. These commitments and arguments serve a role analogous to those posted by law enforcement.

This system does not require commitments to all actions in Figure 1. For example, it only requires a law enforcement agency to commit to a successful request for data (step 3) rather than any proposed request (step 1). The system could easily be augmented with additional commitments and proofs as desired by the court system.

The zero-knowledge arguments about relationships between commitments reveal one additional piece of information. For a law enforcement agency to prove that its committed data request is compatible with a particular court order, it must reveal which specific committed court order authorized the request. In other words, the zero-knowledge arguments reveal the links between specific actions of each party (dashed arrows in Figure 3). These links could be eliminated, reducing visibility into the workflow of surveillance. Instead, entities would argue that their actions are compatible with *some* court order among a group of recent orders.

Remark 4.1. *We now briefly discuss other possible malicious behaviors by law enforcement agencies and companies involving inaccurate logging of data. Though, as mentioned in Remark 3.1, correspondence between real-world events and logged items is not enforceable by technological means alone, we informally argue that our design incentivizes honest logging and reporting of dishonest logging under many circumstances.*

A malicious law enforcement agency could omit commitments or commit to one surveillance request but send the company a different request. This action is visible to the company, which could reveal this misbehavior to the judge. This visibility incentivizes companies to record their actions diligently so as to avoid any appearance of negligence, let alone complicity in the agency's misbehavior.

Similarly, a malicious company might fail to post a commitment or post a commitment inconsistent with its actual behavior. These actions are visible to law enforcement agencies, who could report violations to the judge (and otherwise risk the appearance of negligence or complicity). To make such violations visible to the public, we could add a second law enforcement commitment that acknowledges the data received and proves that it is compatible with the original court order and law enforcement request.

However, even incentive-based arguments do not address the case of a malicious law enforcement agency colluding with a malicious company. These entities could simply withhold from posting any information to the ledger (or post a sequence of false but consistent information), thereby making it impossible to detect violations. To handle this scenario, we have to defer to the legal process itself: when this data is used as evidence in court, a judge should ensure that appropriate documentation was posted to the ledger and that the data was gathered appropriately.

Aggregate statistics. At configurable intervals, the individual courts use MPC to compute aggregate statistics about their surveillance activities.[9] An *analyst*, such as

[9]Microsoft [7] and Google [4] currently release their transparency reports every six months and the Administrative Office of the US Courts does so annually [9]. We take these intervals to be our baseline for the frequency with which aggregate statistics would be released in our system, although releasing statistics more frequently (e.g., monthly) would improve transparency.

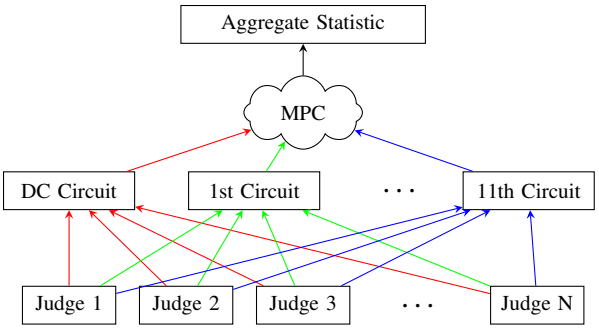


Figure 4: The flow of data as aggregate statistics are computed. Each lower-court judge calculates its component of the statistic and secret-shares it into 12 shares, one for each judicial circuit (illustrated by colors). The servers of the circuit courts then engage in a MPC to compute the aggregate statistic from the input shares.

the Administrative Office of the US Courts, receives the result of this MPC and posts it to the ledger. The particular kinds of aggregate statistics computed are at the discretion of the court system. They could include figures already tabulated in the Administrative Office of the US Court's Wiretap Reports [9] (i.e., orders by state and by criminal offense) and in company-issued transparency reports [4, 7] (i.e., requests and number of users implicated by company). Due to the generality[10] of MPC, it is theoretically possible to compute any function of the information known to each of the judges. For performance reasons, we restrict our focus to totals and aggregated thresholds, a set of operations expressive enough to replicate existing transparency reports.

The statistics themselves are calculated using MPC. In principle, the hundreds of magistrate and district court judges could attempt to directly perform MPC with each other. However, as we find in Section 6, computing even simple functions among hundreds of parties is prohibitively slow. Moreover, the logistics of getting every judge online simultaneously with enough reliability to complete a multiround protocol would be difficult; if a single judge went offline, the protocol would stall.

Instead, we compute aggregate statistics in a hierarchical manner as depicted in Figure 4. We exploit the existing hierarchy of the federal court system. Each of the lower-court judges is under the jurisdiction of one of twelve circuit courts of appeals. Each lower-court judge computes her individual component of the larger aggregate statistic (e.g., number of orders issued against Google in the past six months) and divides it into twelve *secret shares*, sending one share to (a server controlled by) each circuit court of appeals. Distinct shares are represented by separate colors in Figure 4. So long as at least one circuit server remains uncompromised, the lower-court judges can be assured—by the security of the secret-sharing scheme—that their contributions to the larger statistic are confidential. The circuit servers engage in a twelve-party MPC that reconstructs the judges' input data from the shares, computes the desired function, and reveals the result to the analyst. By concentrating the computationally intensive and logistically demanding part of the MPC process in twelve stable servers, this design eliminates many of the performance and reliability challenges of the flat (non-hierarchical) protocol. (Section 6 discusses performance.)

This MPC strategy allows the court system to compute aggregate statistics (towards the *accountability* goal of Section 3.2). Since courts are honest-but-curious, and by the correctness guarantee of MPC, these statistics will be computed accurately on correct data (*correctness* of Section 3.2). MPC enables the courts to perform these computations without revealing any court's internal information to any other court (*confidentiality* of Section 3.2).

4.4 Additional Design Choices

The preceding section described our proposed system with its full range of accountability features. This configuration is only one of many possibilities. Although cryptography makes it possible to release information in a controlled way, the fact remains that revealing more information poses greater risks to investigative integrity. Depending on the court system's level of risk-tolerance, features can be modified or removed entirely to adjust the amount of information disclosed.

Cover sheet metadata. A judge might reasonably fear that a careful criminal could monitor cover sheet metadata to detect surveillance. At the cost of some transparency, judges could post less metadata when committing to an order. (At a minimum, the judge must post the date at which the seal expires.) The cover sheets integral to Judge Smith's proposal were also designed to supply certain information towards assessing the scale of surveillance. MPC replicates this outcome without releasing information about individual orders.

Commitments by individual judges. In some cases, posting a commitment might reveal too much. In a low-crime area, mere knowledge that a particular judge approved surveillance could spur a criminal organization to change its behavior. A number of approaches would address this concern. Judges could delegate the responsibility of posting to the ledger to the same judicial circuits that mediate the hierarchical MPC. Alternatively, each judge could continue posting to the ledger herself, but

[10] *General MPC* is a common term used to describe MPC that can compute arbitrary functions of the participants' data, as opposed to just restricted classes of functions.

instead of signing the commitment under her own name, she could sign it as coming from *some court in her judicial circuit, or nationwide* without revealing which one (*group signatures* [17] or *ring signatures* [33] are designed for this sort of anonymous signing within groups). Either of these approaches would conceal which individual judge approved the surveillance.

Aggregate statistics. The aggregate statistic mechanism offers a wide range of choices about the data to be revealed. For example, if the court system is concerned about revealing information about individual districts, statistics could be aggregated by any number of other parameters, including the type of crime being investigated or the company from which the data was requested.

5 Protocol Definition

We now define a complete protocol capturing the workflow from Section 4. We assume a public-key infrastructure and synchronous communication on authenticated (encrypted) point-to-point channels.

Preliminaries. The protocol is parametrized by:

- a secret-sharing scheme Share,
- a commitment scheme $\mathbb{C}$,
- a special type of zero-knowledge primitive SNARK,
- a multi-party computation protocol MPC, and
- a function CoverSheet that maps court orders to cover sheet information.

Several parties participate in the protocol:

- n judges $J_1, \ldots, J_n$;
- m law enforcement agencies $A_1, \ldots, A_m$;
- q companies $C_1, \ldots, C_q$;
- r trustees $T_1, \ldots, T_r$;[11] and
- P, a party representing the public.
- Ledger, a party representing the public ledger;
- Env, a party called "the environment," which models the occurrence over time of exogenous events.

Ledger is a simple ideal functionality allowing any party to (1) append entries to a time-stamped append-only ledger and (2) retrieve ledger entries. Entries are authenticated except where explicitly anonymous.

Env is a modeling device that specifies the protocol behavior in the context of arbitrary exogenous event sequences occurring over time. Upon receipt of message `clock`, Env responds with the current time. To model the occurrence of an exogenous event e (e.g., a case in need of surveillance), Env sends information about e to the affected parties (e.g., a law enforcement agency).

[11] In our specific case study, $r = 12$ and the trustees are the twelve US Circuit Courts of Appeals. The trustees are the parties which participate in the multi-party computation of aggregate statistics based on input data from all judges, as shown in Figure 4 and defined formally later in this subsection.

Next, we give the syntax of our cryptographic tools,[12] and then define the behavior of the remaining parties.

A **commitment scheme** is a triple of probabilistic polytime algorithms $\mathbb{C} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ as follows.

- $\mathsf{Setup}(1^\kappa)$ takes as input a security parameter κ (in unary) and outputs public parameters pp.
- $\mathsf{Commit}(pp, m, \omega)$ takes as input pp, a message m, and randomness ω. It outputs a commitment c.
- $\mathsf{Open}(pp, m', c, \omega')$ takes as input pp, a message m', and randomness ω'. It outputs 1 if $c = \mathit{Commit}(pp, m', \omega')$ and 0 otherwise.

pp is generated in an initial setup phase and thereafter publicly known to all parties, so we elide it for brevity.

Algorithm 1 Law enforcement agency A_i

- On receipt of a **surveillance request event** $e = (\texttt{Surveil}, u, s)$ **from** Env, where u is the public key of a company and s is the description of a surveillance request directed at u: send message (u, s) to a judge.[13]
- On receipt of a **decision message** (u, s, d) **from a judge** where $d \neq \texttt{reject}$:[14](1) generate a commitment $c = \mathsf{Commit}((s, d), \omega)$ to the request and store (c, s, d, ω) locally; (2) generate a SNARK proof π attesting compliance of (s, d) with relevant regulations; (3) post (c, π) to the ledger; (4) send request (s, d, ω) to company u.
- On receipt of an **audit request** (c, P, z) **from the public**: generate decision $b \leftarrow A_i^{\mathsf{dp}}(c, P, z)$. If $b = \texttt{accept}$, generate a SNARK proof π attesting compliance of (s, d) with the regulations indicated by the audit request (c, P, z); else, send (c, P, z, b) to a judge.[13]
- On receipt of an **audit order** (d, c, P, z) **from a judge**: if $d = \texttt{accept}$, generate a SNARK proof π attesting compliance of (s, d) with the regulations indicated by the audit request (c, P, z).

Agencies. Each agency A_i has an associated decision-making process A_i^{dp}, modeled by a stateful algorithm that maps audit requests to $\texttt{accept} \cup \{0,1\}^*$, where the output is either an acceptance or a description of why the agency chooses to deny the request. Each agency operates according to Algorithm 1, which is parametrized by its own A_i^{dp}. In practice, we assume A_i^{dp} would be instantiated by the agency's human decision-making process.

[12] For formal security definitions, beyond syntax, we refer to any standard cryptography textbook, such as [26].

[13] For the purposes of our exposition, this could be an arbitrary judge. In practice, it would likely depend on the jurisdiction in which the surveillance event occurs, and in which the law enforcement agency operates, and perhaps also on the type of case.

[14] For simplicity of exposition, Algorithm 1 only addresses the case $d \neq \texttt{reject}$, and omits the possibility of appeal by the agency. The algorithm could straightforwardly be extended to encompass appeals, by incorporating the decision to appeal into A_i^{dp}.

[15] This is the step invoked by requests for unsealed documents.

Algorithm 2 Judge J_i

- On receipt of a **surveillance request** (u,s) **from an agency** A_j: (1) generate decision $d \leftarrow J_i^{\mathsf{dp1}}(s)$; (2) send response (u,s,d) to A_j; (3) generate a commitment $c = \mathsf{Commit}((u,s,d),\omega)$ to the decision and store (c,u,s,d,ω) locally; (4) post $(\mathsf{CoverSheet}(d),c)$ to the ledger.
- On receipt of **denied audit request information** ζ **from an agency** A_j: generate decision $d \leftarrow J_i^{\mathsf{dp2}}(\zeta)$, and send (d,ζ) to A_j and to the public P.
- On receipt of a **data revelation request** (c,z) **from the public**:[15] generate decision $b \leftarrow J_i^{\mathsf{dp3}}(c,z)$. If $b = \texttt{accept}$, send to the public P the message and randomness (m,ω) corresponding to c; else, if $b = \texttt{reject}$, send `reject` to P with an accompanying explanation if provided.

Judges. Each judge J_i has three associated decision-making processes, J_i^{dp1}, J_i^{dp2}, and J_i^{dp3}. J_i^{dp1} maps surveillance requests to either a rejection or an authorizing court order; J_i^{dp2} maps denied audit requests to either a confirmation of the denial, or a court order overturning the denial; and J_i^{dp3} maps data revelation requests to either an acceptance or a denial (perhaps along with an explanation of the denial, e.g., "this document is still under seal"). Each judge operates according to Algorithm 2, which is parametrized by the individual judge's $(J_i^{\mathsf{dp1}}, J_i^{\mathsf{dp2}}, J_i^{\mathsf{dp3}})$.

Algorithm 3 Company C_i

- Upon receiving a **surveillance request** (s,d,ω) **from an agency** A_j, if the court order d bears the valid signature of a judge and $\mathsf{Commit}((s,d),\omega)$ matches a corresponding commitment posted by law enforcement on the ledger, then: (1) generate commitment $c \leftarrow \mathsf{Commit}(\delta,\omega)$ and store (c,δ,ω) locally; (2) generate a SNARK proof π attesting that δ is compliant with a s the judge-signed order d; (3) post (c,π) anonymously to the ledger; (4) reply to A_j by furnishing the requested data δ along with ω.

The public. The public P exhibits one main type of behavior in our model: upon receiving an event message $e = (a,\xi)$ from Env (describing either an audit request or a data revelation request), P sends ξ to a (an agency or court). Additionally, the public periodically checks the ledger for validity of posted SNARK proofs, and take steps to flag any non-compliance detected (e.g., through the court system or the news media).

Companies and trustees. Algorithms 3 and 4 describe companies and trustees. Companies execute judge-authorized instructions and log their actions by posting commitments on the ledger. Trustees run MPC to compute aggregate statistics from data provided in secret-shared form by judges; MPC events are triggered by Env.

Algorithm 4 Trustee T_i

- Upon receiving an **aggregate statistic event message** $e = (\texttt{Compute}, f, D_1, \ldots, D_n)$ **from** Env:
  1. For each $i' \in [r]$ (such that $i' \neq i$), send e to $T_{i'}$.
  2. For each $j \in [n]$, send the message (f, D_j) to J_j. Let $\delta_{j,i}$ be the response from J_j.
  3. With parties $T_1, \ldots, T_r$, participate in the MPC protocol MPC with input $(\delta_{1,i}, \ldots, \delta_{n,i})$, to compute the functionality $\mathsf{ReconInputs} \circ f$, where $\mathsf{ReconInputs}$ is defined as follows.

$$\mathsf{ReconInputs}\big((\delta_{1,i'}, \ldots, \delta_{n,i'})\big)_{i' \in [r]} = \big(\mathsf{Recon}(\delta_{j,1}, \ldots, \delta_{j,r})\big)_{j \in [n]}$$

     Let y denote the output from the MPC.[16]
  4. Send y to J_j for each $j \in [n]$.[17]
- Upon receiving an **MPC initiation message** $e = (\texttt{Compute}, f, D_1, \ldots, D_n)$ **from another trustee** $T_{i'}$:
  1. Receive a secret-share $\delta_{j,i}$ from each judge J_j respectively.
  2. With parties $T_1, \ldots, T_r$, participate in the MPC protocol MPC with input $(\delta_{1,i}, \ldots, \delta_{n,i})$, to compute the functionality $\mathsf{ReconInputs} \circ f$.

6 Evaluation of MPC Implementation

In our proposal, judges use secure multiparty computation (MPC) to compute aggregate statistics about the extent and distribution of surveillance. Although in principle, MPC can support secure computation of *any* function of the judges' data, full generality can come with unacceptable performance limitations. In order that our protocols scale to hundreds of federal judges, we narrow our attention to two kinds of functions that are particularly useful in the context of surveillance.

The *extent* of surveillance (totals). Computing totals involves summing values held by the parties without revealing information about any value to anyone other than its owner. Totals become averages by dividing by the number of data points. In the context of electronic surveillance, totals are the most prevalent form of computation on government and corporate transparency reports. How many court orders were approved for cases involving homicide, and how many for drug offenses? How long was the average order in effect? How many orders were issued in California? [9] Totals make it possible to determine the extent of surveillance.

The *distribution* of surveillance (thresholds). Thresholding involves determining the number of data points that exceed a given cut-off. How many courts issued

more than ten orders for data from Google? How many orders were in effect for more than 90 days? Unlike totals, thresholds can reveal selected facts about the distribution of surveillance, i.e., the circumstances in which it is most prevalent. Thresholds go beyond the kinds of questions typically answered in transparency reports, offering new opportunities to improve accountability.

To enable totals and thresholds to scale to the size of the federal court system, we implemented a *hierarchical* MPC protocol as described in Figure 4, whose design mirrors the hierarchy of the court system. Our evaluation shows the hierarchical structure reduces MPC complexity from quadratic in the number of judges to linear.

We implemented protocols that make use of totals and thresholds using two existing JavaScript-based MPC libraries, WebMPC [16, 29] and Jiff [5]. WebMPC is the simpler and less versatile library; we test it as a baseline and as a "sanity check" that its performance scales as expected, then move on to the more interesting experiment of evaluating Jiff. We opted for JavaScript libraries to facilitate integration into a web application, which is suitable for federal judges to submit information through a familiar browser interface, regardless of the differences in their local system setups. Both of these libraries are designed to facilitate MPC across dozens or hundreds of computers; we simulated this effect by running each party in a separate process on a computer with 16 CPU cores and 64GB of RAM. We tested these protocols on randomly generated data containing values in the hundreds, which reflects the same order of magnitude as data present in existing transparency reports. Our implementations were crafted with two design goals in mind:

1. Protocols should scale to roughly 1,000 parties, the approximate size of the federal judiciary [10], performing efficiently enough to facilitate periodic transparency reports.
2. Protocols should not require all parties to be online regularly or at the same time.

In the subsections that follow, we describe and evaluate our implementations in light of these goals.

6.1 Computing Totals in WebMPC

WebMPC is a JavaScript-based library that can securely compute sums in a single round. The underlying protocol relies on two parties who are trusted not to collude with one another: an *analyst* who distributes masking information to all protocol participants at the beginning of the process and receives the final aggregate statistic, and a *facilitator* who aggregates this information together in masked form. The participants use the masking information from the analyst to mask their inputs and send them to the facilitator, who aggregates them and sends the result (i.e., a masked sum) to the analyst. The analyst removes the mask and uncovers the aggregated result. Once the participants have their masks, they receive no further messages from any other party; they can submit this masked data to the facilitator in an uncoordinated fashion and go offline immediately afterwards. Even if some anticipated participants do not send data, the protocol can still run to completion with those who remain.

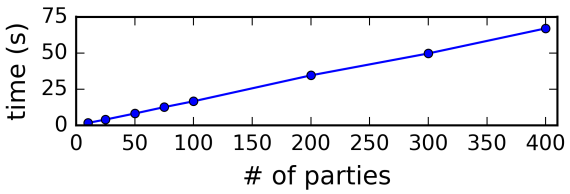


Figure 5: Performance of MPC using WebMPC library.

To make this protocol feasible in our setting, we need to identify a facilitator and analyst who will not collude. In many circumstances, it would be acceptable to rely on reputable institutions already present in the court system, such as the circuit courts of appeals, the Supreme Court, or the Administrative Office of the US Courts.

Although this protocol's simplicity limits its generality, it also makes it possible for the protocol to scale efficiently to a large number of participants. As Figure 5 illustrates, the protocol scales linearly with the number of parties. Even with 400 parties—the largest size we tested—the protocol still completed in just under 75 seconds. Extrapolating from the linear trend, it would take about three minutes to compute a summation across the entire federal judiciary. Since existing transparency reports are typically released just once or twice a year, it is reasonable to invest three minutes of computation (or less than a fifth of a second per judge) for each statistic.

6.2 Thresholds and Hierarchy with Jiff

To make use of MPC operations beyond totals, we turned to Jiff, another MPC library implemented in JavaScript. Jiff is designed to support MPC for arbitrary functionalities, although inbuilt support for some more complex functionalities are still under development at the time of writing. Most importantly for our needs, Jiff supports thresholding and multiplication in addition to sums. We evaluated Jiff on three different MPC protocols: totals (as with WebMPC), *additive thresholding* (i.e., how many values exceeded a specific threshold?), and *multiplicative thresholding* (i.e., did all values exceed a specific threshold?). In contrast to computing totals via summation, certain operations like thresholding require more complicated computation and multiple rounds of communication. By building on Jiff with our hierarchical MPC implementation, we demonstrate that these operations are

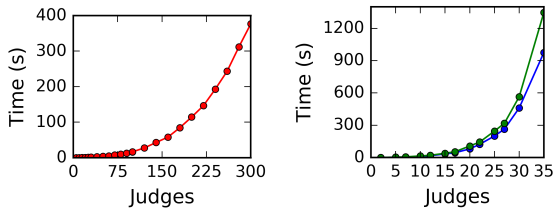


Figure 6: Flat total (red), additive threshold (blue), and multiplicative thresholds (green) protocols in Jiff.

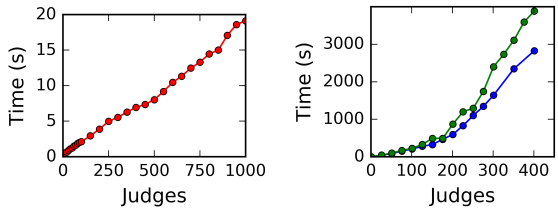


Figure 7: Hierarchical total (red), additive threshold (blue), and multiplicative thresholds (green) protocols in Jiff. Note the difference in axis scales from Figure 6.

viable at the scale required by the federal court system.

As a baseline, we ran sums, additive thresholding, and multiplicative thresholding benchmarks with all judges as full participants in the MPC protocol sharing the workload equally, a configuration we term the *flat* protocol (in contrast to the *hierarchical* protocol we present next). Figure 6 illustrates that the running time of these protocols grows quadratically with the number of judges participating. These running times quickly became untenable. While summation took several minutes among hundreds of judges, both thresholding benchmarks could barely handle tens of judges in the same time envelopes. These graphs illustrate the substantial performance disparity between summation and thresholding.

In Section 4, we described an alternative "hierarchical" MPC configuration to reduce this quadratic growth to linear. As depicted in Figure 4, each lower-court judge splits a piece of data into twelve secret shares: one for each circuit court of appeals. These shares are sent to the corresponding courts, who conduct a twelve-party MPC that performs a total or thresholding operation based on the input shares. If n lower-court judges participate, the protocol is tantamount to computing n twelve-party summations followed by a single n-input summation or threshold. As n increases, the amount of work scales linearly. So long as at least one circuit court remains honest and uncompromised, the secrecy of the lower court data endures, by the security of the secret-sharing scheme.

Figure 7 illustrates the linear scaling of the twelve-party portion of the hierarchical protocols; we measured only the computation time after the circuit courts received all of the additive shares from the lower courts. While the flat summation protocol took nearly eight minutes to run on 300 judges, the hierarchical summation scaled to 1000 judges in less than 20 seconds, besting even the WebMPC results. Although thresholding characteristically remained much slower than summation, the hierarchical protocol scaled to nearly 250 judges in about the same amount of time that it took the flat protocol to run on 35 judges. Since the running times for the threshold protocols were in the tens of minutes for large benchmarks, the linear trend is noisier than for the total protocol. Most importantly, both of these protocols scaled linearly, meaning that—given sufficient time—thresholding could scale up to the size of the federal court system. This performance is acceptable if a few choice thresholds are computed at the frequency at which existing transparency reports are published.[18]

One additional benefit of the hierarchical protocols is that lower courts do not need to stay online while the protocol is executing, a goal we articulated at the beginning of this section. A lower court simply needs to send in its shares to the requisite circuit courts, one message per circuit court to a grand total of twelve messages, after which it is free to disconnect. In contrast, the flat protocol grinds to a halt if even a single judge goes offline. The availability of the hierarchical protocol relies on a small set of circuit courts who could invest in more robust infrastructure.

7 Evaluation of SNARKs

We define the syntax of preprocessing zero-knowledge SNARKs for arithmetic circuit satisfiability [15].

A **SNARK** is a triple of probabilistic polynomial-time algorithms $\mathsf{SNARK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ as follows:

- $\mathsf{Setup}(1^\kappa, R)$ takes as input the security parameter κ and a description of a binary relation R (an arithmetic circuit of size polynomial in κ), and outputs a pair (pk_R, vk_R) of a *proving key* and *verification key*.
- $\mathsf{Prove}(pk_R, (x, w))$ takes as input a proving key pk_R and an input-witness pair (x, w) and outputs a proof π attesting to $x \in L_R$, where $L_R = \{x : \exists w \text{ s.t. } (x, w) \in R\}$.
- $\mathsf{Verify}(vk_R, (x, \pi))$ takes as input a verification key vk_R and an input-proof pair (x, π) and outputs a bit indicating whether π is a valid proof for $x \in L_R$.

Before participants can create and verify SNARKs, they must establish a *proving key*, which any participant can use to create a SNARK, and a corresponding *verification key*, which any participant can use to verify

[18] Too high a frequency is also inadvisable due to the possibility of revealing too granular information when combined with the timings of specific investigations court orders.

a SNARK so created. Both of these keys are publicly known. The keys are distinct for each circuit (representing an NP relation) about which proofs are generated, and can be reused to produce as many different proofs, with respect to that circuit, as desired. Key generation uses randomness that, if known or biased, could allow participants to create proofs of false statements [13]. The key generation process must therefore protect and then destroy this information.

Using MPC to do key generation based on randomness provided by many different parties provides the guarantee that as long as at least one of the MPC participants behaved correctly (i.e., did not bias his randomness, and destroyed it afterward), the resulting keys are good (i.e., do not permit proofs of false statements). This approach has been used in the past, most notably by the cryptocurrency Zcash [8]. Despite the strong guarantees provided by this approach to key generation when at least one party is not corrupted, concerns have been expressed about the wisdom of trusting in the assumption of one honest party in the Zcash setting, which involves large monetary values and a system design inherently centered around the principles of full decentralization.

For our system, we propose key generation be done in a one-time MPC among several of the traditionally reputable institutions in the court system, such as the Supreme Court or Administrative Office of the US Courts, ideally together with other reputable parties from different branches of government. In our setting, the use of MPC for SNARK key generation does not constitute as pivotal and potentially risky a trust assumption as in Zcash, in that the court system is close-knit and inherently built with the assumption of trustworthiness of certain entities within the system. In contrast, a decentralized cryptocurrency (1) must, due to its distributed nature, rely for key generation on MPC participants that are essentially strangers to most others in the system; and (2) could be said to derive its very purpose from not relying on the trustworthiness of any small set of parties.

We note that since key generation is a one-time task for each circuit, we can tolerate a relatively performance-intensive process. Proving and verification keys can be distributed on the ledger.

7.1 Argument Types

Our implementation supports three types of arguments.

Argument of knowledge for a commitment (P_{k}). Our simplest type of argument attests the prover's knowledge of the content of a given commitment c, i.e., that she could open the commitment if required. Whenever a party publishes a commitment, she can accompany it with a SNARK attesting that she knows the message and randomness that were used to generate the commitment. Formally, this is an argument that the prover knows m and ω that correspond to a publicly known c such that $\mathsf{Open}(m,c,\omega)=1$.

Argument of commitment equality (P_{eq}). Our second type of argument attests that the content of two publicly known commitments c_1, c_2 is the same. That is, for two publicly known commitments c_1 and c_2, the prover knows m_1, m_2, ω_1, and ω_2 such that $\mathsf{Open}(m_1,c_1,\omega_1) = 1 \wedge \mathsf{Open}(m_2,c_2,\omega_2)=1 \wedge m_1 = m_2$.

More concretely, suppose that an agency wishes to release relational information—that the identifier (e.g., email address) in the request is the same identifier that a judge approved. The judge and law enforcemnet agency post commitments c_1 and c_2 respectively to the identifiers they used. The law enforcement agency then posts an argument attesting that the two commitments are to the same value.[19] Since circuits use fixed-size inputs, an argument implicitly reveals the length of the committed message. To hide this information, the law enforcement agency can pad each input up to a uniform length.

P_{eq} may be too revealing under certain circumstances: for the public to verify the argument, the agency (who posted c_2) must explicitly identify c_1, potentially revealing which judge authorized the data request and when.

Existential argument of commitment equality ($P_{\exists}$). Our third type of commitment allows decreasing the resolution of the information revealed, by proving that a commitment's content is the same as that of *some* other commitment among many. Formally, it shows that, for publicly known commitments $c, c_1, ..., c_N$ respectively to secret values $(m,\omega),(m_1,\omega_1),...,(m_N,\omega_N)$, $\exists\, i$ such that $\mathsf{Open}(m,c,\omega)=1 \wedge \mathsf{Open}(m_i,c_i,\omega_i)=1 \wedge m=m_i$. We treat i as an additional secret input, so that, for any value of N, only two commitments need to be opened. This scheme trades off between resolution (number of commitments) and efficiency, a question we explore below.

We have chosen these three types of arguments to implement, but LibSNARK supports arbitrary predicates in principle, and there are likely others that would be useful and run efficiently in practice. A useful generalization of P_{eq} and $P_{\exists}$ would be to replace equality with more sophisticated, domain-specific predicates: instead of showing that messages m_1, m_2 corresponding to a pair of commitments are *equal*, one could show $p(m_1,m_2)=1$ for other predicates p (e.g., "less-than" or "signed by same court"). The types of arguments that can be implemented

[19] To produce a proof for P_{eq}, the prover (e.g., the agency) needs to know both ω_2 and ω_1, but in some cases c_1 (and thus ω_1) may have been produced by a different entity (e.g., the judge). Publicizing ω_1 is unacceptable as it compromises the hiding of the commitment content. To solve this problem, the judge can include ω_1 alongside m_1 in secret documents that both parties possess (e.g., the court order).

efficiently will expand as SNARK libraries' efficiency improves; our system inherits such efficiency gains.

7.2 Implementation

We implemented these zero-knowledge arguments with LibSNARK [34], a C++ library for creating general-purpose SNARKs from arithmetic circuits. We implemented commitments using the SHA256 hash function;[20] ω is a 256-bit random string appended to the message before it is hashed. In this section, we show that useful statements can be proven within a reasonable performance envelope. We consider six criteria: the size of the proving key, the size of the verification key, the size of the proof statement, the time to generate keys, the time to create proofs, and the time to verify proofs. We evaluated these metrics with messages from 16 to 1232 bytes on P_{k}, P_{eq}, and $P_{\exists}$ (N = 100, 400, 700, and 1000, large enough to obscure links between commitments) on a computer with 16 CPU cores and 64GB of RAM.

Argument size. The argument is just 287 bytes. Accompanying each argument are its public inputs (in this case, commitments). Each commitment is 256 bits.[21] An auditor needs to store these commitments anyway as part of the ledger, and each commitment can be stored just once and reused for many proofs.

Verification key size. The size of the verification key is proportional to the size of the circuit and its public inputs. The key was 10.6KB for P_{k} (one commitment as input and one SHA256 circuit) and 20.83KB for P_{eq} (two commitments and two SHA256 circuits). Although $P_{\exists}$ computes SHA256 just twice, its smallest input, 100 commitments, is 50 times as large as that of P_{k} and P_{eq}; the keys are correspondingly larger and grow linearly with the input size. For 100, 400, 700, and 1000 commitments, the verification keys were respectively 1.0MB, 4.1MB, 7.1MB, and 10.2MB. Since only one verification key is necessary for each circuit, these keys are easily small enough to make large-scale verification feasible.

Proving key size. The proving keys are much larger: in the hundreds of megabytes. Their size grows linearly with the size of the circuit, so longer messages (which require more SHA256 computations), more complicated circuits, and (for $P_{\exists}$) more inputs lead to larger keys. Figure 8a reflects this trend. Proving keys are largest for $P_{\exists}$ with 1000 inputs on 1232KB messages and shrink as the message size and the number of commitments decrease. P_{k} and P_{eq}, which have simpler circuits, still have bigger proving keys for bigger messages. Although these keys are large, only entities that create each kind of proof need to store the corresponding key. Storing one key for each type of argument we have presented takes only about 1GB at the largest input sizes.

Key generation time. Key generation time increased linearly with the size of the keys, from a few seconds for P_{k} and P_{eq} on small messages to a few minutes for $P_{\exists}$ on the largest parameters (Figure 8b). Since key generation is a one-time process to add a new kind of proof in the form of a circuit, we find these numbers acceptable.

Argument generation time. Argument generation time increased linearly with proving key size and ranged from a few seconds on the smallest keys to a couple of minutes for largest (Figure 8c). Since argument generation is a one-time task for each surveillance action and the existing administrative processes for each surveillance action often take hours or days, we find this cost acceptable.

Argument verification time. Verifying P_{k} and P_{eq} on the largest message took only a few milliseconds. Verification times for $P_{\exists}$ were larger and increased linearly with the number of input commitments. For 100, 400, 700, and 1000 commitments, verification took 40ms, 85ms, 243ms, and 338ms on the largest input. These times are still fast enough to verify many arguments quickly.

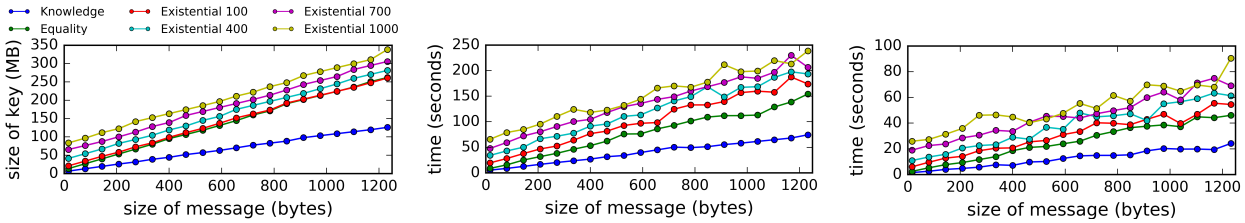
8 Generalization

Our proposal can be generalized beyond ECPA surveillance to encompass a broader class of *secret information processes*. Consider situations in which independent institutions need to act in a coordinated but secret fashion and, at the same time, are subject to public scrutiny. They should be able to convince the public that their actions are consistent with relevant rules. As in electronic surveillance, accountability requires the ability to attest to compliance without revealing sensitive information.

Example 1 (FISA court). Accountability is needed in other electronic surveillance arenas. The US Foreign Intelligence Surveillance Act (FISA) regulates surveillance in national security investigations. Because of the sensitive interests at stake, the entire process is overseen by a US court that meets *in secret*. The tension between secrecy and public accountability is even sharper for the FISA court: much of the data collected under FISA may stay permanently hidden inside US intelligence agencies, while data collected under ECPA may eventually be used in public criminal trials. This opacity may be justified, but it has engendered skepticism. The public has no way of knowing what the court is doing, nor any means of assuring itself that the intelligence agencies under the authority of FISA are even complying with the rules of that

[20] Certain other hash functions may be more amenable to representation as arithmetic circuits, and thus more "SNARK-friendly." We opted for a proof of concept with SHA256 as it is so widely used.

[21] LibSNARK stores each bit in a 32-bit integer, so an argument involving k commitments takes about $1024k$ bytes. A bit-vector representation would save a factor of 32.



(a) Proving key size. (b) Key generation time. (c) Argument generation time.

Figure 8: SNARK evaluation

court. The FISA court itself has voiced concern about that it has no independent means of assessing compliance with its orders because of the extreme secrecy involved. Applying our proposal to the FISA court, both the court and the public could receive proofs of documented compliance with FISA orders, as well as aggregate statistics on the scope of FISA surveillance activity to the full extent possible without incurring national security risk.

Example 2 (Clinical trials). Accountability mechanisms are also important to assess behavior of private parties, e.g., in clinical trials for new drugs. There are many parties to clinical trials and much of the information involved is either private or proprietary. Yet, regulators and the public have a need to know that responsible testing protocols are observed. Our system can achieve the right balance of transparency, accountability and respect for privacy of those involved in the trials.

Example 3 (Public fund spending). Accountability in spending of taxpayer money is naturally a subject of public interest. Portions of public funds may be allocated for sensitive purposes (e.g., defense/intelligence), and the amounts and allocation thereof may be publicly unavailable due to their sensitivity. Our system would enable credible public assurances that taxpayer money is being spent in accordance with stated principles, while preserving secrecy of information considered sensitive.

8.1 Generalized Framework

We present abstractions describing the generalized version of our system and briefly outline how the concrete examples fit into this framework. A *secret information process* includes the following components.

- A set of *participants* interact with each other. In our ECPA example, these are judges, law enforcement agencies, and companies.
- The participants engage in a *protocol* (e.g., to execute the procedures for conducting electronic surveillance). The protocol messages exchanged are hidden from the view of outsiders (e.g., the public), and yet it is of public interest that the protocol messages exchanged adhere to certain rules.
- A set of *auditors* (distinct from the *participants*) seeks to audit the protocol, by verifying that a set of *accountability properties* are met.

Abstractly, our system allows the controlled disclosure of four types of information.

Existential information reveals the existence of a piece of data, be it in a participant's local storage or the content of a communication between participants. In our case study, existential information is revealed with commitments, which indicate the existence of a document.

Relational information describes the actions participants take in response to the actions of others. In our case study, relational information is represented by the zero-knowledge arguments that attest that actions were taken lawfully (e.g., in compliance with a judge's order).

Content information is the data in storage and communication. In our case study, content information is revealed through aggregate statistics via MPC and when documents are unsealed and their contents made public.

Timing information is a by-product of the other information. In our case study, timing information could include order issuance dates, turnaround times for data request fulfilment by companies, and seal expiry dates.

Revealing combinations of these four types of information with the specified cryptographic tools provides the flexibility to satisfy a range of application-specific accountability properties, as exemplified next.

Example 1 (FISA court). *Participants* are the FISA Court judges, the agencies requesting surveillance authorization, and any service providers involved in facilitating said surveillance. The *protocol* encompasses the legal process required to authorize surveillance, together with the administrative steps that must be taken to enact surveillance. *Auditors* are the public, the judges themselves, and possibly Congress. Desirable *accountability properties* are similar to those in our ECPA case study: e.g., attestations that certain rules are being followed in issuing surveillance orders, and release of aggregate statistics on surveillance activities under FISA.

Example 2 (Clinical trials). *Participants* are the institutions (companies or research centers) conducting clinical trials, comprising scientists, ethics boards, and data

analysts; the organizations that manage regulations regarding clinical trials, such as the National Institutes of Health (NIH) and the Food and Drug Administration (FDA) in the US; and hospitals and other sources through which trial participants are drawn. The *protocol* encompasses the administrative process required to approve a clinical trial, and the procedure of gathering participants and conducting the trial itself. *Auditors* are the public, the regulatory organizations such as the NIH and the FDA, and possibly professional ethics committees. Desirable *accountability properties* include, e.g., attestations that appropriate procedures are respected in recruiting participants and administering trials; and release of aggregate statistics on clinical trial results without compromising individual participants' medical data.

Example 3 (Public fund spending). *Participants* are Congress (who appropriates the funding), defense/intelligence agencies, and service providers contracted in the spending of said funding. The *protocol* encompasses the processes by which Congress allocates funds to agencies, and agencies allocate funds to particular expenses. *Auditors* are the public and Congress. Desirable *accountability properties* include, e.g., attestations that procurements were within reasonable margins of market prices and satisfied documented needs; and release of aggregate statistics on the proportion of allocated money used and broad spending categories.

9 Conclusion

We present a cryptographic answer to the accountability challenge currently frustrating the US court system. Leveraging cryptographic commitments, zero-knowledge proofs, and secure MPC, we provide the electronic surveillance process a series of scalable, flexible, and *practical* measures for improving accountability while maintaining secrecy. While we focus on the case study of electronic surveillance, these strategies are equally applicable to a range of other *secret information processes* requiring accountability to an outside auditor.

Acknowledgements

We are grateful to Judge Stephen Smith for discussion and insights from the perspective of the US court system; to Andrei Lapets, Kinan Dak Albab, Rawane Issa, and Frederick Joossens for discussion on Jiff and WebMPC; and to Madars Virza for advice on SNARKs and LibSNARK.

This research was supported by the following grants: NSF MACS (CNS-1413920), DARPA IBM (W911NF-15-C-0236), SIMONS Investigator Award Agreement Dated June 5th, 2012, and the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370.

References

[1] Bellman. `https://github.com/ebfull/bellman`.

[2] Electronic Communications Privacy Act. 18 USC 2701 et seq.

[3] Foreign Intelligence Surveillance Act. 50 U.S.C. ch. 36.

[4] Google transparency report. `https://www.google.com/transparencyreport/userdatarequests/countries/?p=2016-12`.

[5] Jiff. `https://github.com/multiparty/jiff`.

[6] Jsnark. `https://github.com/akosba/jsnark`.

[7] Law enforcement requests report. `https://www.microsoft.com/en-us/about/corporate-responsibility/lerr`.

[8] Zcash. `https://z.cash/`.

[9] Wiretap report 2015. `http://www.uscourts.gov/statistics-reports/wiretap-report-2015`, December 2015.

[10] ADMINSTRATIVE OFFICE OF THE COURTS. Authorized judgeships. `http://www.uscourts.gov/sites/default/files/allauth.pdf`.

[11] ARAKI, T., BARAK, A., FURUKAWA, J., LICHTER, T., LINDELL, Y., NOF, A., OHARA, K., WATZMAN, A., AND WEINSTEIN, O. Optimized honest-majority MPC for malicious adversaries - breaking the 1 billion-gate per second barrier. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017* (2017), IEEE Computer Society, pp. 843–862.

[12] BATES, A., BUTLER, K. R., SHERR, M., SHIELDS, C., TRAYNOR, P., AND WALLACH, D. Accountable wiretapping–or–i know they can hear you now. *NDSS* (2012).

[13] BEN-SASSON, E., CHIESA, A., GREEN, M., TROMER, E., AND VIRZA, M. Secure sampling of public parameters for succinct zero knowledge proofs. In *Security and Privacy (SP), 2015 IEEE Symposium on* (2015), IEEE, pp. 287–304.

[14] BEN-SASSON, E., CHIESA, A., TROMER, E., AND VIRZA, M. Scalable zero knowledge via cycles of elliptic curves. *IACR Cryptology ePrint Archive 2014* (2014), 595.

[15] BEN-SASSON, E., CHIESA, A., TROMER, E., AND VIRZA, M. Succinct non-interactive zero knowledge for a von neumann architecture. In *23rd USENIX Security Symposium (USENIX Security 14)* (San Diego, CA, Aug. 2014), USENIX Association, pp. 781–796.

[16] BESTAVROS, A., LAPETS, A., AND VARIA, M. User-centric distributed solutions for privacy-preserving analytics. *Communications of the ACM 60*, 2 (February 2017), 37–39.

[17] CHAUM, D., AND VAN HEYST, E. Group signatures. In *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings* (1991), D. W. Davies, Ed., vol. 547 of *Lecture Notes in Computer Science*, Springer, pp. 257–265.

[18] DAMGÅRD, I., AND ISHAI, Y. Constant-round multiparty computation using a black-box pseudorandom generator. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings* (2005), V. Shoup, Ed., vol. 3621 of *Lecture Notes in Computer Science*, Springer, pp. 378–394.

[19] DAMGÅRD, I., KELLER, M., LARRAIA, E., PASTRO, V., SCHOLL, P., AND SMART, N. P. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings* (2013), J. Crampton, S. Jajodia, and K. Mayes, Eds., vol. 8134 of *Lecture Notes in Computer Science*, Springer, pp. 1–18.

[20] FEIGENBAUM, J., JAGGARD, A. D., AND WRIGHT, R. N. Open vs. closed systems for accountability. In *Proceedings of the 2014 Symposium and Bootcamp on the Science of Security* (2014), ACM, p. 4.

[21] FEIGENBAUM, J., JAGGARD, A. D., WRIGHT, R. N., AND XIAO, H. Systematizing "accountability" in computer science. Tech. rep., Yale University, Feb 2012. Technical Report YALEU/DCS/TR-1452.

[22] FEUER, A., AND ROSENBERG, E. Brooklyn prosecutor accused of using illegal wiretap to spy on love interest, November 2016. https://www.nytimes.com/2016/11/28/nyregion/brooklyn-prosecutor-accused-of-using-illegal-wiretap-to-spy-on-love-interest.html.

[23] GOLDWASSER, S., AND PARK, S. Public accountability vs. secret laws: Can they coexist?: A cryptographic proposal. In *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society, Dallas, TX, USA, October 30 - November 3, 2017* (2017), B. M. Thuraisingham and A. J. Lee, Eds., ACM, pp. 99–110.

[24] JAKOBSEN, T. P., NIELSEN, J. B., AND ORLANDI, C. A framework for outsourcing of secure computation. In *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security, CCSW '14, Scottsdale, Arizona, USA, November 7, 2014* (2014), G. Ahn, A. Oprea, and R. Safavi-Naini, Eds., ACM, pp. 81–92.

[25] KAMARA, S. Restructuring the nsa metadata program. In *International Conference on Financial Cryptography and Data Security* (2014), Springer, pp. 235–247.

[26] KATZ, J., AND LINDELL, Y. *Introduction to modern cryptography*. CRC press, 2014.

[27] KROLL, J., FELTEN, E., AND BONEH, D. Secure protocols for accountable warrant execution. 2014. `http://www.cs.princeton.edu/felten/warrant-paper.pdf`.

[28] LAMPSON, B. Privacy and security usable security: how to get it. *Communications of the ACM 52*, 11 (2009), 25–27.

[29] LAPETS, A., VOLGUSHEV, N., BESTAVROS, A., JANSEN, F., AND VARIA, M. Secure MPC for Analytics as a Web Application. In *2016 IEEE Cybersecurity Development (SecDev)* (Boston, MA, USA, November 2016), pp. 73–74.

[30] MASHIMA, D., AND AHAMAD, M. Enabling robust information accountability in e-healthcare systems. In *HealthSec* (2012).

[31] PAPANIKOLAOU, N., AND PEARSON, S. A cross-disciplinary review of the concept of accountability: A survey of the literature, 2013. Available online at: `http://www.bic-trust.eu/files/2013/06/Paper_NP.pdf`.

[32] PEARSON, S. Toward accountability in the cloud. *IEEE Internet Computing 15*, 4 (2011), 64–69.

[33] RIVEST, R. L., SHAMIR, A., AND TAUMAN, Y. How to leak a secret. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings* (2001), C. Boyd, Ed., vol. 2248 of *Lecture Notes in Computer Science*, Springer, pp. 552–565.

[34] SCIPR LAB. libsnark: a C++ library for zkSNARK proofs. `https://github.com/scipr-lab/libsnark`.

[35] SEGAL, A., FEIGENBAUM, J., AND FORD, B. Privacy-preserving lawful contact chaining: [preliminary report]. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society* (New York, NY, USA, 2016), WPES '16, ACM, pp. 185–188.

[36] SEGAL, A., FORD, B., AND FEIGENBAUM, J. Catching bandits and only bandits: Privacy-preserving intersection warrants for lawful surveillance. In *FOCI* (2014).

[37] SMITH, S. W. Kudzu in the courthouse: Judgments made in the shade. *The Federal Courts Law Review 3*, 2 (2009).

[38] SMITH, S. W. Gagged, sealed & delivered: Reforming ecpa's secret docket. *Harvard Law & Policy Review 6* (2012), 313–459.

[39] SUNDARESWARAN, S., SQUICCIARINI, A., AND LIN, D. Ensuring distributed accountability for data sharing in the cloud. *IEEE Transactions on Dependable and Secure Computing 9*, 4 (2012), 556–568.

[40] TAN, Y. S., KO, R. K., AND HOLMES, G. Security and data accountability in distributed systems: A provenance survey. In *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on* (2013), IEEE, pp. 1571–1578.

[41] VIRZA, M., November 2017. Private communication.

[42] WANG, X., RANELLUCCI, S., AND KATZ, J. Global-scale secure multiparty computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017* (2017), B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds., ACM, pp. 39–56.

[43] WEITZNER, D. J., ABELSON, H., BERNERS-LEE, T., FEIGENBAUM, J., HENDLER, J. A., AND SUSSMAN, G. J. Information accountability. *Commun. ACM 51*, 6 (2008), 82–87.

[44] XIAO, Z., KATHIRESSHAN, N., AND XIAO, Y. A survey of accountability in computer networks and distributed systems. *Security and Communication Networks 9*, 4 (2016), 290–315.