My research focuses on improving the security of distributed systems with multiple administrative domains. Today, many large computer systems are no longer controlled by an individual administrator or a single organization; rather, they consist of multiple subsystems that are interconnected but each belong to a different organization. Typical examples of systems in this category include the Internet, many web services, and most cloud computing platforms.

When a system has multiple administrative domains, it is much more difficult to ensure that it works correctly. Individual components can malfunction for many reasons, ranging from benign faults (such as misconfigurations or hardware failures) to malicious attacks in which an adversary takes complete control over some machines and then tries to subvert the system from within. Since each domain can only see the state of the machines under its own control, it can be very challenging to even detect the presence of such a problem; diagnosing and repairing the problem can be even more difficult. As an additional complication, there are scenarios where the domains have different and potentially conflicting goals. In these scenarios, the domains can be reluctant to share information with each other, or they may even have an incentive to prevent a successful diagnosis.

My work has pioneered accountability as a new approach to this problem. This approach is inspired by the offline world, where we have been successfully operating large “distributed systems” – such as governments or international organizations – for a long time. An accountable distributed system can detect when some part of it is malfunctioning or has been compromised, and it can produce irrefutable evidence that proves the existence of the problem to a third party, such as the administrator of another subsystem. I have also been exploring secure provenance as a way to diagnose compromised subsystems and to perform forensics, and I have been leveraging differential privacy to provide a way for the domains to safely share a limited amount of information with each other, without the risk of accidentally leaking sensitive information.

I consider systems to be my core area, and a number of my results have been published at the top systems conferences, such as SOSP [17, 35], OSDI [3, 12, 24] and NSDI [1, 6, 13, 18]. However, my work also has strong connections to security/privacy and networking. I enjoy collaborating both within and across disciplines – for instance, I have been working with database researchers on applying big-data concepts to network security, with privacy experts on developing new data-mining technologies that can work across domains, and with members of the PL community on building tools that can make data sharing safe for non-experts. As a result, my work has also appeared at conferences in other areas, such as networking [7, 11, 30, 31, 33], security [2, 19, 21], databases [22, 34, 36], and measurement [8, 9, 14, 32].

An important part of my work is building and evaluating real systems. This is useful to demonstrate that a new idea, algorithm, or technique can be used in a complete system, that it solves a real problem, and that it can be deployed in practice. I also look for opportunities to collaborate with industry: for instance, my group and I have ongoing projects with Akamai, Intel, and Microsoft Research, while Facebook and Google have partially funded our work.

1 Accountability

The primary focus of our work has been on developing novel techniques and algorithms that add accountability to distributed systems. Accountability adds a new dimension to the existing work on dependable systems, which focuses mostly on fault prevention. Accountability does not attempt to prevent faults, nor does it try to conceal the symptoms of faults that cannot be prevented. Instead, it ensures that faults can be reliably detected and linked to a faulty node. Once the system is aware of the fault, it can reconfigure itself to exclude the faulty node, or it can report the fault to an administrator, who can then correct it [16]. This is similar to the way we use accountability in the offline world, e.g., in governments, banks, or other large organizations; hence the name.

Foundations: The basis of this work was my dissertation research on PeerReview [17], an algorithm that can enforce accountability in any distributed system, provided that the system can be described as a collection of deterministic
state machines. PeerReview was the first algorithm to offer accountability guarantees for arbitrary applications, rather than for an individual application. PeerReview can provably detect a very general class of faults [15], including any fault that causally affects at least one correct node. PeerReview increases a system’s robustness because designers do not have to predict (and possibly be wrong about) which faults might occur in their system, nor do they have to tailor countermeasures for each new application. Thus, PeerReview provides a ‘safety net’ that catches any fault, whether it is caused by hardware malfunction, misconfiguration, manipulation by users or operators, or by a deliberate attack.

**Generalizations:** In the work that followed, my group and I have generalized the approach in several dimensions. The first generalization involved **nondeterministic actions:** the initial algorithm assumed that the behavior of each node would be deterministic, but in practice, many systems use some amount of randomness, such as coin tosses, which cannot easily be verified after the fact. CSAR [2] addresses this by providing strong, accountable randomness based on cryptographic techniques. The second generalization extended our approach to work with **unmodified software.** Our initial accountability techniques required changes to the system’s source code, but this is not always an option – for instance, a cloud platform might wish to run arbitrary software binaries for its customers. To address this challenge, we developed accountable virtual machines [12], which can work with arbitrary binaries. Third, we have developed ways to work with **weak identities and partial views,** which can occur when there are parts of the system that are simply not observable by other domains [1]. And finally, we have been working on ways to provide accountability in systems with **private information** that cannot be revealed to a third party. NetReview [13] first addressed this challenge by verifying partial specifications of the expected behavior, which can exclude the private information; a later system, SPIDeR [33], provided even stronger privacy guarantees by leveraging a specialized, highly efficient type of zero-knowledge proof. NetReview and SPIDeR were specialized solutions for BGP interdomain routing, but, in our ongoing work, we are generalizing this approach to other applications based on a combination of zero-knowledge proofs and epistemic reasoning [27].

**Practical applications:** We have applied accountability to a variety of different systems over the years, including a server-based network file system, a decentralized email system, a large-scale content distribution system, two multiplayer games [12], access networks [20], and the Internet’s interdomain routing system [11, 13, 33]. Also, in collaboration with Akamai Technologies, we have recently applied accountability to Akamai’s NetSession system, a peer-assisted content distribution network with currently over 24 million clients [1, 32]. The wide variety of use cases demonstrates that accountability is widely applicable. Also, in each system, our techniques were able to detect a number of different problems that had been previously described in the literature; this shows that accountability is a general technique, and that it supersedes many specific countermeasures that have been proposed for particular problems. Finally, our evaluation of each system showed that accountability is practical, both in terms of its run-time overhead and in terms of the effort required to deploy it.

**Extension to time-critical systems:** In our ongoing work, we are extending accountability to **temporal misbehavior** — that is, to systems where problems can not only occur when a machine “does the wrong thing”, but also when it “does the right thing at the wrong time”. For instance, this is important in cyber-physical systems, such as factory control systems, where a command given at the wrong time can result in considerable physical damage. We have developed a technique called time-deterministic replay (TDR) [3] that can detect problems of this type, and we have shown that this approach has another surprising benefit: it can be used to detect a broad class of covert timing channels — including novel and unknown channels! Defending systems against covert channels has been a long-standing problem in computer security, and the best known techniques were only effective against specific known types of channels, so our approach represents a big step forward. In a project that is sponsored jointly by Intel and NSF, we are also working a way to provide not only detection but also guaranteed, automatic recovery [5].

## 2 Secure forensics

Accountability can help to detect problems in large distributed systems, but it cannot provide a detailed diagnosis. To determine exactly what went wrong, and to assess what needs to be repaired, operators additionally need a way to perform **secure forensics.** At a high level, the operator should be able to ask the system to explain a particular symptom – perhaps a suspicious entry in a routing table, or an odd entry in a certain file. However, if the system has been compromised, how can the operator be sure that the system is not telling lies, or returns a forged but plausible explanation?

**Foundations:** We have been developing **secure network provenance** [35], a forensic capability for distributed systems that can give strong guarantees **even if** part of the system is controlled by a malicious adversary. Our approach is based
on the concept of data provenance from the database community, which we have substantially extended and adapted for use in adversarial environments. Our techniques can securely track the provenance of data as it flows through a distributed system; if a compromised machine attempts to lie about an explanation, our approach ensures that the lie can be detected and attributed to that machine.

We have developed a new provenance model for secure forensics [34], and we have designed an algorithm that can securely and efficiently maintain forensic data and answer forensic queries. We have proven the correctness of our algorithm, and, to demonstrate its practicality, we have implemented it in a system called SNooPy [35]. SNooPy has been applied to three applications: interdomain routing, a distributed hashtable, and Hadoop MapReduce.

**Remembering past events:** In a system like SNooPy, it is not enough to keep the provenance of data that currently exists in the system, since this would provide an attacker with a way to cover his tracks by deleting telltale information. To address this, we have developed a time-aware provenance model [36] that can efficiently maintain forensic data about both current and past events. Since this approach requires the system to store a considerable amount of data, we have developed several different ways to encode it, depending on the characteristics of the application and the workload it is handling. We have also designed a system called DistTape [36] that can predict the cost of different encodings, and adaptively choose the best encoding for a given deployment scenario.

**Handling negative symptoms:** While developing SNooPy, we realized that most existing forensic techniques focus on investigating “positive” symptoms, such as an unexpected event that the operator has observed. But what if the problem is that an expected event was not observed? To address this surprisingly common scenario, we have developed a generalization we call negative provenance [30], which can explain the absence of an expected event using a form of counterfactual reasoning. We have implemented our approach in a system called Y! [31], and we have shown that Y! can help administrators to quickly diagnose problems in software-defined networks.

**Simplifying the forensic output:** Secure provenance can provide a comprehensive explanation of any event that occurred in the system, but in some cases this explanation is too comprehensive, and the operator may need to spend a considerable amount of time on pinpointing the actual root cause. In our current work, we are developing a way to simplify the provenance using reference events – that is, events that are not related to the attack but are otherwise very similar to the event of interest. We have found that, by reasoning about the structural differences between the provenances of the two events, we can provide the operator with far simpler explanations, and sometimes even identify the root cause directly [4].

**Automatic repair:** We are currently working on an extension that would allow us to not only diagnose problems but also repair them automatically. To this end, we are generalizing the concept of provenance – which has typically been applied only to data – to also cover program code. We call this approach meta-provenance [29]. Just like standard provenance can tell us which data was used to generate an observed message or other output, meta-provenance can tell us which parts of a program were involved. Using this information, we can then automatically generate very targeted program changes that can repair the problem. We are currently evaluating this approach in the context of software-defined networks.

## 3 Differential privacy

In systems with multiple administrative domains, it is often necessary to share some information across domain boundaries – for instance, to detect subtle attacks. However, domains that work with private information are sometimes reluctant to share information in this way, since this creates a risk of accidental data leaks. Recent privacy scandals, such as the ones involving Netflix and AOL, have shown that this concern is well-founded, and that leaks can occur even when a domain is very careful to protect its data. Thus, if we want to deploy security techniques (such as accountability) that require cross-domain data sharing, we must be able to offer the domains very strong privacy guarantees.

We have been developing techniques that can provide such guarantees using differential privacy, a new kind approach to data privacy that was developed in the theory community. Differential privacy builds on a solid mathematical foundation and offers very strong, provable privacy guarantees, even under worst-case conditions. However, it can be difficult to deploy even in a centralized setting because of a number of practical challenges: for instance, it is not always easy to tell for a non-expert whether a particular type of data can be shared safely in this model.

**Practical tools for non-experts:** To address these challenges, we started by developing technologies that can make differential privacy easier to use. In collaboration with PL experts at Penn, we have built a programming language that can be used to describe queries over private data sets, as well as special type system that can be used to prove that
answering such queries would maintain differential privacy [10, 19, 28]. Thus, if a domain wants to share some data with another domain, all the operator has to do is describe the data in our programming language and verify that the resulting program is accepted by the compiler. If so, the data is safe to share. This is easy to do for anyone with a bit of programming experience – they do not need to be privacy experts!

Another practical challenge with differential privacy is that a crucial parameter, $\varepsilon$, is difficult to choose correctly, particularly for non-experts. To address this, we developed an economic model that can be used to estimate suitable values for $\varepsilon$ [21]. This model also revealed some interesting facts about differential privacy; for instance, we found that offering strong privacy guarantees can actually reduce (and not increase) the cost of certain types of studies.

**Protection against side channels:** If the domains exchange data automatically, without a human in the loop, there is a risk that private information might leak through side channels – even if the data itself is differentially private. For instance, a malicious domain might ask another domain for some data and then measure the time it takes to generate a response; small variations in this time can be surprisingly disclosing. To protect against such information leaks, we developed a technique we call predictable transactions [19], and we showed that, in combination with differential privacy, our technique can completely eliminate an important class of side channels. This is a substantial improvement over prior work, most of which can only mitigate side channels but not eliminate them.

**Efficient distributed queries:** Next, we developed techniques for combining data across domains. Most of the prior work on differential privacy assumed that the private data is stored in a central database; however, in a setting with multiple domains, it would be better if each domain could keep its data under its own control, and to answer “global” queries (i.e., ones that require data from more than one domain) with a distributed protocol. We initially developed a technique called DJoin [24] that can efficiently answer queries involving a certain type of cross-domain join. DJoin was orders of magnitude faster than previous approaches, and was the first technique to work at practical time scales. In our ongoing work, we are generalizing the technology behind DJoin to other kinds of queries, as well as to other computation models, including graph algorithms [25].

**Practical applications:** So far, we have applied our techniques in two scenarios. The first was publications – that is, medical studies – that are based on sensitive data. In this case, it is often difficult to balance the privacy concerns of the study participants against the need to include enough details to make the study repeatable. To address this, we developed a system called VerDP [23] that uses a combination of differential privacy and noninteractive zero-knowledge proofs to offer both strong privacy and strong verifiability.

Our second application scenario is systemic risk in financial networks. Briefly, systemic risk measures the likelihood of a “snowball effect” in the banking system, where the initial failure of a few banks causes financial trouble at more and more banks, until the system as a whole is in danger of collapsing. (This effect is thought to have been partly responsible for the 2008 financial crisis.) Economists have found ways to detect this risk early enough to prevent a collapse, but it requires access to the banks’ sensitive financial data. We have built a system called DStress [25, 26] that can perform the necessary computations efficiently and with strong privacy guarantees. DStress is orders of magnitude faster than competing technologies: it can do in hours what would otherwise have taken hundreds of years.

**References**


