Association for Computing Machinery
Advancing Computing as a Science & Profession

Ubiquity, an ACM publication
July 2017

Mixing Computation with People

An Interview with Marianne Winslett

by Richard T. Snodgrass

Editor's Introduction

In this interview, we learn about five fascinating subjects: security in manufacturing, negotiating trust in the web, updating logical databases, differential privacy, and scientific computing (including its security issues). This is a confluence that has, at its roots, the thorny problems that arise when you mix computation with people. Some beautiful technical results, many originated by Marianne Winslett, now address those challenges, but some surprises crop up along the way.

Mixing Computation with People

An Interview with Marianne Winslett

by Richard T. Snodgrass

Richard Snodgrass: Today I want to explore with you a variety of technical topics that you have been confronting over your 30-plus year career in computer science research.

It seems to me, a core impulse of your research is the interaction between computer science applications and the blood-and-guts humans who create and update these applications. That interaction can elicit very interesting technical issues.

The first topic I would like to explore is how do we protect manufacturing from being attacked?

Marianne Winslett: Until recently, the special security needs of the manufacturing community had escaped the notice of the computer security research community, which is unfortunate since our factories are already under heavy attack today. The driving factor behind these attacks is for high value products, like airplanes or pharmaceuticals, it can be quite a bit cheaper and faster to steal the intellectual property (IP) associated with those products than to try to come up with that IP on one's own. Here IP means two different things. First, there's the design of the product. And second, quite separately, there's the process used to fabricate that design. Each manufacturer has its own secret sauce, a way to make parts faster, cheaper, or better than its competitors. That's the manufacturer's competitive advantage.

RS: Would these manufacturers have patents on these processes?

MW: If you file for a patent, you have to reveal what you're patenting. I would think many manufacturers would prefer to keep their processes secret instead. The Coca-Cola formula is a trade secret because that way you don't have to divulge the formula.

RS: What would be an example of a high-value product exhibiting this problem?

MW: One example is the key parts of an airplane, like its turbines. My favorite story to illustrate the prevalence of IP theft is the first time the F-35 airplane took a test flight; its entire operating system and all the data readings had been stolen by the time it landed.

RS: How do they know it was stolen?

MW: Maybe they observed unauthorized transmissions of data during flight. Or maybe it's an urban legend. All I know for sure is Edward Snowden released documents that mention the F-35 blueprints were stolen. Then competitors came up with airplane designs that looked quite a bit like the F-35, without having to spend the billions of dollars necessary to figure out how to design and manufacture it.

RS: That's astounding to me given the F-35 is a military fighter jet. And so I would think the military would have all sorts of procedures for ensuring this didn't happen.

MW: Yes, but when you make an airplane, there may be a thousand vendors involved, making all sorts of different parts, and it's hard to protect the information in such an enormous supply chain. My colleague Bill King tells me about 50 of those parts are the critical ones, the things whose designs and manufacturing processes you really don't want to get stolen. The turbine blade shape is one of those secrets. When installed in an airplane, those 50 critical parts are physically surrounded by maybe 500 other parts, so that if you stole the details for enough of the surrounding parts, you could reconstruct some of those critical parts.

Among the thousand of companies making the parts for the plane, there's a long tail of small manufacturers that have vital know-how about how to manufacture important parts better than other companies do. So you can't drop those small manufacturers from the manufacturing supply chain for the plane. But these small manufacturers, these mom and pop places, they're not IT experts. They're quite vulnerable to theft, and not just from the computers in their front office. Factories typically have PCs out on the factory floor, and those PCs, by policy, are never patched or upgraded.

RS: Is this an operational problem in that we could have operational changes to solve it? Or does this have some deep computer science aspects to it?

MW: I would say both things are true. It's true we need to come up with a better kind of software system controlling the information flow and processes on factory floors. And it's a research area that definitely needs to be looked into, something that's futuristic and forward looking.

But, also, at the same time, there are reasons for everything that's done on a factory floor. There's a reason they never patch and never upgrade. It's because any time you do that, you're at significant risk of downtime. As computer scientists, we can appreciate the fact that if you upgrade one little thing somewhere or patch one little thing, it can make everything else break. And a factory that's running 24/7 really can't afford the downtime that comes with computer system failure. So there's an "if it ain't broke, don't fix it" attitude.

To make things worse, when you buy factory floor machinery nowadays, it typically comes with a PC embedded in it. I've seen an $800,000 brand new machine that's running Windows XP internally. And it will run Windows XP until the day the machinery is scrapped. If you buy really high quality factory machinery, it can last 30 years, and it will forever be running what it was running on the day that you bought it. Long before that day, Microsoft will have stopped supporting that operating system, so there won't be any patches available for newly identified vulnerabilities. So you couldn't patch it even if you wanted to. And even before that, the computer might not be powerful enough to run the latest version of the operating system and other software.

RS: Is one of the problems that it's even running a large operating system rather than a very small, specialized system?

MW: Even if the manufacturer buys a dumb machine that just has firmware on it, the firmware can be attacked. Even a machine from the 1970s, with just a programmable logical controller, we know how to attack those if we so choose.

RS: Is this on the radar screen for funding agencies?

MW: Bill King and I have a grant now from the Department of Homeland Security to see how to help make the manufacturing sector more secure. It's a particularly critical problem right now, because manufacturers realize that if they collect and save a lot more information about a product throughout its lifecycle, they can shorten the time necessary to create new versions of it. For example, they can analyze information from maintenance records and use it to make the

product more maintainable, more usable, and so on. So they see real competitive advantages in storing and making use of a lot more information.

But they're also terrified at the potential attack avenues that having this information easily available will open up, and rightfully so.

RS: Is the security community, in general, working on this really important problem?

MW: I'd say it's not really on people's radar screens. In part, that might be because the manufacturing sector did try to sweep it under the rug, rather like the electrical power sector used to try to believe, or pretend, they didn't face major security challenges. This is a natural human tendency. But I think things are changing now. Manufacturers are becoming more willing to admit the magnitude of the problem.

We do need some forward looking, "rethink everything from the ground up" approaches to factory floor systems. But, on the other hand, there are some really interesting things you can do to protect legacy equipment. If you respect the fact that a factory machine could last for 50 years, rather like a space craft that's shot off by NASA—it is what it is, you've got to live with it—then what can you do to protect that machine? That's an interesting research problem in and of itself. So in our project, we've been looking at a lot of possibilities for protecting, in a respectful way, these legacy machines with their old hardware and software.

RS: Can you give us a hint as to how one might do that?

MW: One example is manufacturers are afraid of having their factory floor accessible over the internet because that could open the factory floor up to attack. So instead, they put the recipes for what the factory floor machines are supposed to be doing onto USB memory sticks, flash drives, and bring them to the factory floor that way. And they have no idea these flash drives are a great vector for attack. They think using USB drives is safer than using the internet.

When I heard that—and given that even the most expensive, modern factory floor equipment made today comes with a USB interface—it just made me smack my head as a security researcher. I can't believe they're doing this and they think it's safe and they trust these USB drives. Manufacturers have heard of Stuxnet, and it alarms them—as it should—but still somehow they don't realize how vulnerable they are and that just preventing outside access to the machines over the internet doesn't cut it. To be protected, they have to do more.

A similar problem occurs when you have to call outside vendors to repair factory equipment. Every minute the factory line is shut down you're losing money. The repair person shows up, and maybe you know the person and trust them, maybe you've never seen them before. Either way, the repair person walks up to the machine and sticks a USB into it. Heaven knows where that USB has been. It horrifies me to think of it.

The first step in changing that is to make manufacturing people aware of the risks associated with using USB memory sticks, and the second step is to offer them an easy-to-use, low-cost alternative. For the first step, there's a consortium called the Digital Manufacturing and Design Innovation Institute up in Chicago that has beautiful new factory equipment, and brings together many people across the manufacturing sector. With that kind of audience in mind, we are putting together a demonstration of various ways these machines can be attacked through USBs, explained in a way we think will make sense to a manufacturing expert who is not a computer scientist. We need to show them you just can't trust people's USBs, and it's easy to introduce self-replicating malware onto your PC and have it spread across the factory floor. [Note: Videos of those attacks are now up on YouTube.]

As for the second step, offering manufacturers a good alternative: You can already buy protective devices intended to guard against USB malware, but they don't offer what is needed in a manufacturing environment. One of our entrepreneurially minded team members, Avesta Hojjati, has teamed up with another professor to address that problem, and I hope they come up with a great solution.

RS: So these are just some kind of simple things that might help a lot, along with realization of the attack vectors that might be present. What are some of the more computer science research questions?

MW: The whole problem of how to protect USB drives is much broader than a manufacturing concern. Attacks on programmable logic controllers (PLCs) are kind of hot in the security research community right now. People are looking at them because of the internet of things (IoT), and the manufacturing sector is just one example of an existing internet of things. [Editor's Note: since we spoke, the first really big attack exploiting IoT took place on September 20, 2016.] Any attack you can do on IoT you can probably also do on the factory floor (although they're not exactly the same, for example, because of the importance of legacy hardware and software in factories).

In a factory, or anywhere else, you need to be able to protect your firmware. Even if you're

allowing firmware upgrades, if you're allowing vendor upgrades, there needs to be some form of attestation that will let you trust that upgrade. That's an interesting issue that's been getting a lot of attention in the past few years for the case of advanced metering infrastructure in the power grid industry. Of course, the power grid world and the manufacturing world aren't exactly the same. But as with all cyberphysical systems, there are overlaps in what the security issues are and in potential solution approaches.

Another concern is researchers haven't really explored all of the additional ways you can attack a factory floor. Our most recent piece of work is about the case where your smartphone is near a factory floor machine, and whether you know it or not, your phone is recording the sound the machine makes and perhaps recording the information from some of its other built-in sensors. For a 3-D printer and a CNC mill, we showed recording is enough to allow a reconstruction of the design and manufacturing process for the object being fabricated. In other words, just the sound the machine makes is enough to figure out what the factory is doing.

RS: Wow. It's amazing that the audio can provide that level of information.

MW: It could even be that you get a phone call while you're standing next to the machine. If the caller records the background audio, it can tell them what's being made and how it's being made. [Editor's Note: Since we spoke, this attack was published in CCS'16, as was a related attack from another group.]

If I were an attacker, though, I don't think I'd have to resort to a phone-based attack to steal manufacturers' IP. I'd try to break into the factory's network, from the comfort of my own office. If that didn't work, I'd try to break into one of the factory's repair vendors, and take advantage of the holes that the factory opens to give those repair people remote access when things break down. Time is money for an attacker, so I'd take the easiest route.

In our research project, our goal is to raise the cost of stealing manufacturing designs and processes, which currently is quite low. And theft is the number one threat, but it's not the only threat. As an attacker, you could introduce flaws that damage factory machines, your competitors' machines. You could make them self-destruct, as StuxNet did. Once the equipment is physically damaged, it can take a very long time to repair or replace it. Or you could introduce faults into the designs or the recipe so that the objects that were manufactured had flaws that were going to make them break.

RS: One thing I always think about when I hear about research on new attacks, when smart researchers figure out really interesting attacks that weren't perhaps known before, and publish details on them, does that make the situation worse?

MW: It can. And for that reason, security researchers usually give the vendor a chance to come up with a patch before they announce a new vulnerability in the vendor's product. Of course, that courtesy won't help if the intended victims don't install the patch, or if no one can come up with a patch, or if the product is no longer supported.

Another consideration is that any new vulnerability we researchers think of may already be available for sale as a zero-day exploit. And probably a number of spy agencies in the world already know about the vulnerability, too. Neither of those groups is going to publicize the problem. So it's probably just as well if the vulnerability comes to public attention, so that it can be patched.

For example, there's a very powerful USB attack called "BadUSB." Its publication spurred vendors to come up with a more secure design for the firmware inside of new USBs. The vendors would never have done that work if they hadn't realized how vulnerable every USB's internal code is to attack. And the essential concept behind BadUSB has been turned into a nice penetration- testing tool by another entrepreneurial soul. Still, BadUSB can be used to do nasty things.

So publication of new attacks is both good and bad. It's always an arms race in the security world.

RS: Let's shift gears. One of the words I associate with you in my mind is "trust," in two senses. First, you are a very trustworthy, honest person. And second, you have made many contributions in the area of "trust."

One of your most referenced papers has the evocative title, "Negotiating Trust in the Web" (in *IEEE Internet Computing,* 2002). I'm reminded of the prevalent cartoon of a dog typing at his (her?) computer with the caption, "On the Internet, nobody knows you're a dog." In what specific way is the web/internet different in terms of the fundamental notion of "trust"?

MW: The web added an interesting new wrinkle to the problem of authorization. In the olden days, before the internet was very popular, inside a computer system, you were always doing business with users who you knew. You knew what their privileges should be. You knew what

they should be allowed to do and not allowed to do. It was a closed system, no strangers allowed.

The web changed that very quickly. All of a sudden, users whom you'd never heard of before were coming to your web hosts and asking for services. And so the question became: How do you decide whether a stranger is authorized to do an action that they ask to do? That dynamic problem didn't really exist before.

The web was our first enormous open system. It opened up some interesting new directions of research for authorization, including issues associated with privacy for the users who were asking for authorization. The move to open systems spawned a whole new kind of work in the community, a shift from a focus on authentication—who are you—to more subtle issues.

RS: What is your sense as to the situation today with regard to this problem?

MW: Most popular systems today are relying on one authority that knows the user well to share information about that user with other places. Sometimes this sharing is benign, and sometimes it isn't. A benign example is Eduroam [a secure, world-wide roaming access service developed for the international research and education community], which you probably have seen when you visited other universities. If I'm not mistaken, Eduroam is a Shibboleth-style system that exploits the fact that your home institution knows a lot of your attributes, including the key fact that you are a member of their community. Your home institution uses its knowledge of your attributes to help other institutions decide whether you should be authorized to do something there.

The Eduroam model is a good model. It doesn't involve your home institution sharing everything it knows about you with other places. But out in the commercial world, you see a lot of requests for you to log in with Facebook, or Google Plus.

RS: Or LinkedIn.

MW: That's not nearly so benign because there's always a commercial incentive to share a lot of data about you so that advertising can be targeted more precisely to your exact behavior. Asking you to log in with Facebook is a way to collect, in one logically centralized place, information about everything you've ever done on the internet and then use that for marketing purposes.

RS: How is your behavior different from, say, your daughter's because of your security and privacy research?

MW: First, I never, ever log into anywhere with Facebook or any other site. I never use those kinds of setups. Never.

Second, my phone has almost no apps. Every app you put on your phone is an additional attack surface, and your phone is as vulnerable as its most vulnerable app. Apps can't be properly vetted for security vulnerabilities in the App Store. There aren't enough people to do that, and the automated techniques aren't quite there yet to do it in a very thorough way. So every time you add an app, you're just adding another potentially open door for someone to attack your phone. My phone has very few apps.

My browser has almost no plugins or extensions. Exact same story there. In fact, we won a prize at Usenix Security some years ago, for a paper about tools for automatically looking for security vulnerabilities in browser add-ons. Every toolbar, every plugin, and every add-on you put on your browser, it can't be thoroughly checked for vulnerabilities. It's just not possible. So I don't use them.

RS: What does a normal user do? Do they download a lot of apps and plugins?

MW: Oh, absolutely, totally, yes. That's totally normal behavior to have a phone that's packed with apps. And each one is a security risk and also a privacy risk, because a lot of them do track various sorts of behavior of yours. To me, it's not worth it.

RS: So have you impressed upon your daughter this concern?

MW: Not so much as I've just mentioned that it exists, because my daughter is naturally an extremely cautious person. And, in fact, I never said to myself that I'm not going to put these things on my computer and phone. But since I was part of a research project that was demonstrating how vulnerable these things actually are, I just naturally found myself not wanting to add them, unless they had a really high value for me personally, which is occasionally the case. But I bet my daughter doesn't have nearly as many apps on her phone as most people do, just from comments I've made at home.

RS: Is this a failure of our community (computer science researchers and practitioners) to protect the public, given that we develop the technology of plugins and apps to begin with?

MW: One of the reasons code has bugs is that developers are kind of optimists. They're thinking about a particular use case, and they're thinking what code they can build to make this use case work. They aren't really thinking about all of these extraordinary, unusual circumstances that don't match that use case and could cause the code to fail. And in particular, developers are not very good at thinking like attackers. Actually, they're no good at that at all. It's a specialized skill, and when you're training someone to be a security researcher, the first thing they have to learn is to think like an attacker. Attacking is all about not following the usual use case and looking for sneaky little openings that a developer probably didn't think about. For an attacker, nothing is cheating.

A trivial example is buffer overflow. Buffer overflows create vulnerabilities that good attackers can exploit. But as you work to create a new piece of functionality, how often do you think about the fact that a printf statement could have a buffer overflow? Not very often.

Maybe we have a social responsibility to train future software engineers and software designers to think like attackers, so they can design and build systems a little more defensively. If you don't know how an attacker thinks, it's very hard for you to recognize and close a lot of the gaps that an attacker might try to exploit. So maybe we have a social responsibility to quash this natural optimism. On the other hand, no system can be invulnerable because every system has bugs, and bugs are potential avenues for attack.

RS: It's kind of a frustrating situation. It doesn't seem like there are easy answers anywhere here.

MW: I agree. No easy solutions.

RS: Let's talk about another area of research, one you started right after your doctorate, that of updating logical databases.

MW: My background was in databases, but I found myself hanging around with AI people who were worried about the problem of how to update a description of the state of the world in response to new information**.**

RS: What would be a simple example?

MW: One classical, ancient example from the domain of non-monotonic and counterfactual reasoning would be if I told you, "Matilda is my favorite bird." If you heard that, you would probably have a mental model of a wing-flapping, beak-pecking, egg-laying creature. But then if I add that you can read about her at angrybirds.com, then you would revise your view of the world to reflect the fact that Matilda probably actually doesn't fly. She is probably a cartoon bird. Humans have no problem with absorbing new information that contradicts what they believed before, but contradictions make a mess out of traditional logical reasoning.

RS: Here's some new information about my world: It was sunny when we started talking, but now it's raining outside.

MW: If we imagine you have a set of logic formulas that describes how the world is, you're going to revise those formulas to reflect the fact that it's now raining when it wasn't before. But when you make that change, you don't want to change the completely unrelated parts of your beliefs about the world. For example, you don't also decide your spouse has taken a flight to New Guinea.

Hence, everything else should more or less remain unchanged. You want to change your set of beliefs as little as possible to accommodate the fact that it's now raining. Your changes would only include small things like there are going to be puddles and people might be using umbrellas. Little changes.

When I first because aware of this problem, people were taking the exact formulas in these theories at face value and treating them as first-class objects. So for example, if your set of beliefs had included the formula "It's not raining right now and my spouse is downstairs making breakfast," the approaches to revising those beliefs that people were looking at the time would treat that quite differently from the case where your beliefs included the two formulas "It's not raining right now" and "My spouse is downstairs making breakfast." Even though, from a mathematical point of view, one formula or two simpler formulas are just different representations of the same information.

I realized the key difficulty was that the way people had proposed to revise a set of beliefs was quite dependent on the way those beliefs were represented. That meant their revisions would produce very different results depending on the exact formulas used to represent the beliefs. That sensitivity was bad, because there are many equivalent ways to represent the same

underlying reality. Putting it another way, the logic formulas used to represent beliefs had semantics, i.e., an underlying (mathematical) reality, and we needed to define what it meant to change a set of beliefs with respect to the semantics, i.e., the underlying reality, not with respect to the specific syntactic representation used to represent that reality.

RS: What are the relevant approaches now for belief maintenance?

MW: Back in those days, people thought we were going to be able to conquer all sorts of reasoning problems in AI using mathematical logic. But it turned out to be really hard to represent the way humans think using logic. Statistical and probabilistic approaches have shown a lot more promise in a lot of areas and have become the dominant paradigm now in AI.

But to the extent that things *can* be represented in logic, I'd say everybody understands now that when you want to revise a set of beliefs to be consistent with a new observation, then those revisions must defined with respect to the set of possible worlds—realities—that are represented by your formulas.

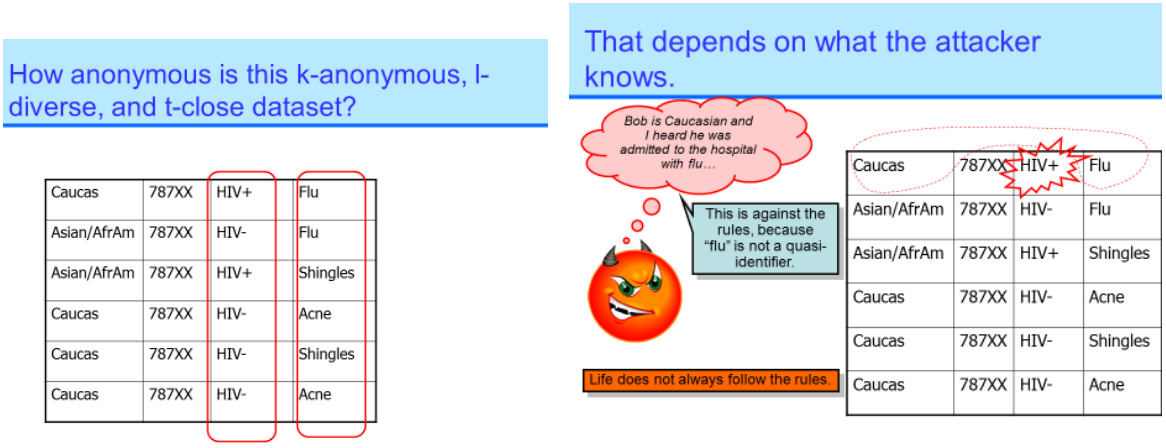
At their core, I think probabilistic and statistical approaches are consistent with a logic-based approach. But the key contribution of the newer probabilistic and statistical approaches is in how they assign probabilities to states of the world. And that's really important. That was needed. And people hadn't really gone in that direction yet, back in the 1980s.

RS: Is there a connection with trust in this regard?

MW: Not in the security sense of the word *trust* in our conversation earlier, though it's true that belief and trust are interrelated, and probability is relevant for both of them. When you're trying to do common sense reasoning, you're not really very worried about the corner cases. You're just trying to figure out what the most likely world is and gracefully handle contradictory information that comes in. Whereas in security, you have to be quite concerned about those low probability worlds because an attacker can take advantage of the loopholes in those situations and sneak into your system. So it's almost the opposite.

RS: When you were in Singapore, you headed a large project on differential privacy. Some of our readers might be more familiar with privacy-preserving methods such as "k-anonymity." How do these two compare, and how do they differ?

MW: K-anonymity is a feel-good privacy measure that ordinary humans can understand: The personally identifiable information in my records will be indistinguishable from that of *k* other people. But the reality is measures like k-anonymity don't guarantee anything about privacy. Researchers kept identifying holes in k-anonymity and subsequent proposed approaches, and proposing ways to patch those holes, without end. For example, here's a nice little example (given below), I think originally from Vitaly Shmatikov, that's k-anonymous, l-diverse, and t-close, but you can still figure out exactly who is HIV positive. So these measures don't give you any real privacy guarantees.



On the other hand, k-anonymity and similar measures are better than doing nothing, because they do raise the cost of a successful attack. The attacker does have to do some mental gymnastics to figure out which person a record is about.

Differential privacy is different in the sense that it gives a firm upper bound on the chance that an attacker can figure out whether a particular individual is included in the data set or not. It works by adding noise to the results of an analysis. The fact that differential privacy gives you specific provable numerical guarantee is lovely. But it's not a cure-all for a number of reasons.

The first problem has to do with accuracy. When you add noise to an analysis result to make it differentially private, you can measure how much accuracy you lose. If you add noise using the vanilla method, *usually* you lose so much accuracy the results aren't useful for their intended purpose. And that's why there's so much research on this topic. If you're very clever, you may be able to come up with a new method of making analysis results differentially private while preserving a lot more accuracy than the vanilla method does. And that's great, when it works.

Another problem is differential privacy is easy to abuse when deployed in the field. For example, there's this notion of a privacy budget for the dataset, and each analysis you do chips away at the budget. No one understands how to set the budget in the first place, i.e., how

much privacy is enough. And when the day arrives that you've used up the budget, you should stop doing new analyses of the dataset, forever. Do you think people will have the discipline to stop analyzing their data when the budget runs out? Analyses of overlapping data owed by different organizations can chip away invisibly at each other's budgets.

For example, maybe you want to produce a report every month about the prevalence of HIV on each block of your city. De facto, the data sets for each month overlap heavily. Every single month, you're going to leak a little bit more info, you're going to use up a little more of your budget. Maybe after you've done this for five years the budget is gone, and if you publish one more report, the privacy guarantee won't hold anymore.

Another problem is differential privacy only guarantees that it will be hard to tell whether a particular person was included in the dataset, and that's not always the information that needs to be kept private. As an extreme example, a differentially private analysis concludes 99 percent of the people on my block are currently infected with the Zika virus, then it doesn't matter whether I was included in the study or not. Everybody is going to know I am probably also infected with the Zika virus, so my blood can potentially transmit Zika to other people until my infection runs its course. Differential privacy can't wave a magic wand and preserve my privacy here, given the statistical conclusions that came out of the study.

Differential privacy is an endless, bottomless pit for research because for every single kind of analysis that you can imagine anyone doing on any dataset, maybe if you are clever enough, you can come up with a way to preserve enough accuracy for differentially private results to still be accurate enough to be useful.

Our research project focused on differential privacy for biomedical data. If I summarize the entire project, I would say differential privacy is great when it works, that is, when we can preserve enough accuracy for the analysis results to still be useful for their intended purpose. But right now, many analyses you want to do couldn't be accurate enough with differential privacy, or you would run into problems because you need to do the analysis repeatedly over the same individuals. So, in sum, when it works, it's wonderful, and the rest of the time, it's useless except as a good research problem.

RS: You mentioned it might be possible to come up with a method where you could do a monthly report pretty much forever and not use up your budget. Are there any negative results in this area of analysis, accuracy versus differential privacy that say that something *cannot* be done?

MW: There are some negative results, yes, and some beliefs in the theory community about what is not possible. There's also been a fair bit of wishful thinking about what differential privacy can do, much of which was summarized in a nice 2011 paper called "No Free Lunch in Data Privacy" by Daniel Kifer and Ashwin Machanavajjhala.

We actually hit up against one of the theoreticians' beliefs in our project. Suppose you wanted to make a dataset differentially private. That's not something people normally do, because you lose so much accuracy. Instead, they just make the result of an analysis differentially private. But suppose you wanted to make a dataset differentially private. We thought we could retain more accuracy if we first compressed the data using compressive sensing, which retains its essential features, and then added noise to it. We were right about the improved accuracy, by a significant factor, and theoreticians got excited about that. But you still lose so much accuracy that our technique isn't practical.

RS: In your intuition, do you think this is going to be like other parts of the security arms race? Or do you think there is a possibility that some magic thing like public key cryptography, which seems to do a lot of things right, will emerge?

MW: Even public key crypto is an arms race, at a minimum in terms of key length. As computers get more and more powerful, we can break things that were encrypted long ago with short keys. So there is an arms race even there. It's just a slow motion arms race because it's based on how fast computers get faster. So could there be some magic secret sauce in differential privacy? I think there could be. But don't ask me what the secret sauce is. I would love to know.

RS: What kind of training should someone have to work on differential privacy?

MW: First, you really need to understand probability and statistics. Second, differential privacy is very technically challenging. It's easy to come up with what you think are clever ways to make your analysis more accurate but still differentially private, while in fact you made a mistake, and the results are no longer differentially private.

A few years back, when a lot of differential privacy papers started to be submitted to conferences, our team was asked to review a lot of them. Most of these papers had technical errors in them, fundamental flaws, and that's not what you normally find with submissions to major conferences. We saw very smart people who are well known in other areas make fatal errors when they were new to differential privacy. One good way to avoid that is to collaborate

with someone who has already worked in differential privacy, who has a trained eye and can tell you cheated when you did this or that, and you don't meet the guarantee of privacy anymore. Xiaokui Xiao filled that guru role in our project.

If you want to use other people's published results in differential privacy, then you need to be extremely careful! The area is so technically challenging you should *not* assume that published techniques are correct, even in top-tier conferences, even for papers with well-known authors from other areas of research. I am sorry to say as of 2016, incorrect papers are still appearing in top venues. There aren't enough referees yet with a trained eye for this tricky topic.

RS: Does one need domain knowledge like bioinformatics for this? Or is it mainly the statistics and probability?

MW: As a referee, you just need an excellent grasp of the relevant statistics, and what to watch out for in terms of violations of the differential privacy guarantee. As a researcher, if you're theoretically inclined, then the same is true because you're trying to come up with results that aren't specific to any particular domain. But in our project, we were focusing on biomedical data, so we also needed to work with a biostatistician who knew what types of analyses were the most important in that area, where privacy was most needed, and what level of accuracy was needed. They were mostly interested in various types of regression, and in genome-wide association surveys.

In genomic research, their goal was to be able to share analysis results openly and freely without having to go through an IRB [institutional review board] process to address privacy concerns. They wanted to accelerate research in genomic and other types of biomedical research by not having to worry about privacy issues in the analysis results. They wanted to be able to publish their results on the web and not worry.

RS: This sounds, in part, more like statistics research than computer science research.

MW: It's very statistical, as are some kinds of data mining. But the kind of clever tricks you need to come up with to retain accuracy, that's kind of a puzzle solving, traditional computer science type of problem. We computer scientists are good at solving puzzles.

RS: So a variety of skills are needed in this. And it's a really important problem.

MW: A good non-biomedical example of its importance is census records. Census data products are very important for the economy. Lots of businesses buy them and use them to figure out what they should be selling where. It's been known for a very long time that these data products leak private information. The Census Bureau realized they might be able to make some of their data products, in other words, the results of analyses, differentially private and then sell them without having to feel bad about the fact that they were leaking information about some individuals. Now they sponsor research in this area. For example, they sponsored an early project by Johannes Gehrke and his colleagues on making the analysis of commute data in Boston differentially private, so the data product wouldn't leak so much information about where individuals worked and lived.

RS: It strikes me that the original use of punch cards was in the census of 1890, and we're still having data issues with the census 130 years later.

MW: Yeah, it's tough.

RS: But, obviously, important. The census was societally important even back then. You've got another big research area in scientific data management. What role, if any, did the presence of the National Center for Supercomputing Application (NCSA)—which is, of course, at the University of Illinois—play in your getting into this topic?

MW: There's so much use of parallel computing at Illinois that I've come to believe any computer scientist who is there long enough will eventually do or be influenced by parallel computing. Throughout the College of Engineering, so many scientists and engineers use parallel computing to do enormous simulations: the weather, or the Big Bang, or fluid flow in a rocket, or whatever interests them. Parallel computing is a very common tool on our campus. The most recent project I did in this area was tons of fun, a data science project about I/O usage on supercomputers.

In supercomputer design, I/O is the poor second cousin to computation, because the designers focus on getting as much computing power as possible. There's also a frustrating gap between the advertised peak performance of the I/O system and what people actually get when they write out files from an application. To help figure out what was going on, we had a data science project that analyzed and visualized the data collected by Darshan, a very lightweight I/O monitoring application from Argonne National Laboratory, which runs on a number of major

supercomputer installations. The main focus in the project was on figuring out what, out of all of this logged data, was meaningful to show to people, how to characterize it, and how to visualize it for them so they could just look at a graph and understand what was going on with their application or their job or their platform. That required a lot of domain knowledge, which, fortunately, we already had.

We were shocked by the results. For example, we found most jobs using POSIX or MPI-IO got less I/O bandwidth than you'd get with four modern USB flash drives. The users are on a supercomputer, and they're getting four thumb drives of I/O bandwidth! It was just ridiculous.

RS: Where is all of that bandwidth being lost? I mean, the hardware has, as you say, tremendous I/O bandwidth. You're not getting it at the end of the application. Is it the operating system? Is it the API? Where was the bottleneck?

MW: The bandwidth is lost all along the line. Anywhere you can imagine it being lost, some application is losing it there. The funniest example: You'd be amazed how many supercomputer users out there were writing out their data with fprintf statements, which is about as slow as you can get. We found a quarter of the jobs on a major platform were completely bypassing the I/O facilities intended for them, and were instead using functions like fprintf and fscanf. Of course those users were getting abysmal performance. It was shocking because nobody knew they were doing this. That's the glory of a data science project like that, you can discover what you never even imagined was going on. In our case, that included users who were not really prepared to run on a supercomputer and didn't understand that that's not the right way to do I/O. It's fantastic you can use data science techniques to pick out applications that are doing their I/O in a very naive way, and then suggest to their developers they attend I/O boot camp. When the camp instructors explain how to get good I/O performance, the users see it's not really harder than using fprintf statements. The developers are happier, the system administrators are happier, everyone benefits.

I came into the project with a lot of preconceived ideas about what the focus of research on parallel I/O should be. Those myths got busted pretty quickly. Instead of focusing on how to make the peak I/O performance even higher, we need to be thinking about how to ensure that everyone reaches a certain minimum level of performance. If every job got the equivalent of four USB flash drives of throughput (1 GB/sec), then an enormous amount of system resources would be freed up.

Another preconceived idea of mine that got busted was my beliefs about how parallel I/O should be performed. There are three major paradigms for parallel I/O, and we discovered none of them guarantee good I/O throughput. Perhaps more surprising, none of them guarantee *bad* I/O throughput either (fprintf doesn't count as a paradigm). For example, we saw many jobs that wrote out millions of files. Millions, just for one single job, if you can believe that. And users typically run their applications over and over again, with each and every run generating millions of files. We thought this is just so bone-headed! But, in fact, some of the users who did that got great throughput. More generally, we found for every major paradigm for parallel I/O, the devil is in the details. You might get really good I/O throughput or not, depending on how you tune it. It's kind of unfortunate because, for example, if you move to another platform, it's unlikely your I/O will work at the same throughput as on your previous platform. It's probably going to need to be retuned.

Our results were very warmly received by the parallel computing community. My graduate student who did the project enjoyed it so much she changed from being a parallel computing person to a data science person—a convert! It's rare for a researcher to have clients eagerly awaiting her results. The project was enormous fun and, I hope, eye opening for the people who design supercomputers as well as the people who administer and use them.

RS: I can see you as the host of a Myth Busters TV show.

MW: I never thought of that, but it's true that by applying data science to log data, you can bust a lot of myths by discovering what's actually happening as opposed to what everybody thought was happening.

RS: Yeah. I think it would be a fun show, too. Look at this fprintf: aha!

Getting on the data side as contrasted with the I/O side, you've written several papers about multidimensional array storage and access within scientific computations. In your view, are issues with such arrays now pretty well understood? Or are we still in the Wild West with respect to multidimensional array storage and access?

MW: Same thing as above: you can do it well, or you can do it horribly.

RS: Where else in supercomputing scientific data management are we in the Wild West, in your experience, if any?

MW: In some parts of computer science, you have to develop a certain mindset to be able to do the work. One of those is security. You have to be able to think like an attacker, which is a skill we talked about before, one that you have to actually acquire because it's not natural for most system-builders. As an engineer, we want to build things up. We don't want to figure out how to pull one little pin and have the whole thing come tumbling down. So you have to develop that attacker mindset. And in parallel computing, you have to develop a parallel computing mindset, which really is very different from sequential computing.

This is a problem because if you've got a fantastic sequential code and you want to run it on an enormous dataset or over some huge span of simulated time, like a fine-grained simulation of the entire evolution of the universe, then you have to learn quite a bit about parallel computing to turn your sequential application into a parallel one. And it's a little bit of a Wild West, because you can't just wave a wand and figure out how to make that application run fast in parallel.

That means it's not like the database world, where we were pretty good at figuring out how to make stuff run in parallel when we needed to, whether it was parallel versions of relational databases or map reduce. Parallel computing for scientific problems isn't like that. It's not a black art, but it's an art form. So I wouldn't exactly call it the Wild West, but it's definitely untamed.

RS: If scientific data management is challenging, and security and in particular trust management is challenging, and both of them require their own mindsets, what happens when you bring them together? I'm thinking about your paper on negotiating trust on the grid you wrote back in 2005. Is the combination even harder?

MW: Not really, because security is fairly orthogonal to parallelism in scientific computing, thank heavens. They're at different granularities, the job level versus what happens inside the job. So when you combine them, the problem doesn't really get worse. What made security get harder was when we went to a distributed open system like the web, and started doing business with individuals we didn't really know. But making it parallel doesn't really make it harder.

RS: Okay. So that's more a security problem than a parallel problem.

MW: They're orthogonal, yeah.

RS: Well, that's a ray of hope there then.

MW: But wait! Supercomputers have always been very attractive targets for attack.

RS: Oh, no…

MW: Always. Maybe in the olden days, the goal was to brag that you broke into one of the world's fastest computers. But now, they're attractive for a different reason. The attackers run jobs to try to forge bitcoins.

RS: I thought the supercomputers were all behind these big, locked doors, and you had to have a special physical key to even get in the room.

MW: Nope. They're shared because they're national or at least organization-wide resources. People need to be able to access them remotely from the other side of the world or, at least, the other side of the country. That means if I'm using a supercomputer in San Diego, and someone compromises my account back in Illinois, then they may be able to log in as me and use my time allocation on that computer. So at supercomputing sites, the administrators are always watching for strange activity. Having the same user logged in at two different physical locations is a great example. When they see something like that, the administrators call you up and ask you what's going on.

RS: Thank you so much for your time.

MW: It was my pleasure.

This interview has been condensed and edited for clarity.

Biographies

Richard Snodgrass is a member of the *Ubiquity* Editorial Board. He is also a Professor of Computer Science at the University of Arizona, working in the areas of ergalics and temporal databases.

Since 1987, Marianne Winslett has been a professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign, where she founded the DAIS (Data and Information Systems) research group. After four years as the director of Illinois's research center in Singapore, the Advanced Digital Sciences Center, she returned to the U.S. in 2013. She is an ACM Fellow and the recipient of an ACM SIGMOD Contributions Award, as well as a Presidential Young Investigator Award from the U.S. National Science Foundation. Her research specialties include information security and management of scientific data. She holds a Ph.D. from Stanford University. Winslett is the former vice-chair of ACM SIGMOD and the co-editor-in-chief of ACM *Transactions on the Web*, and has served on the editorial boards of ACM *Transactions on Database Systems*, IEEE *Transactions on Knowledge and Data Engineering*, ACM *Transactions on Information and Systems Security*, the *Very Large Data Bases Journal*, and ACM *Transactions on the Web*. She has received two best paper awards for research on managing regulatory compliance data (VLDB, SSS), one best paper award for research on analyzing browser extensions to detect security vulnerabilities (USENIX Security), and one for keyword search (ICDE).

DOI: 10.1145/3105917

