

COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

10/2020 VOL.63 NO.10

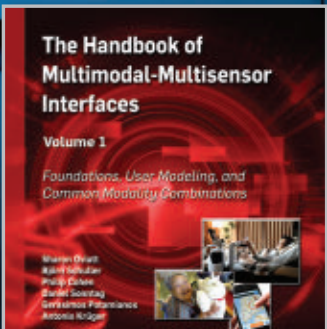
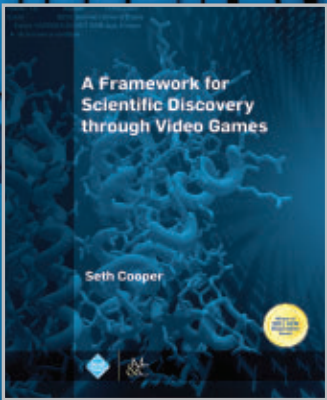
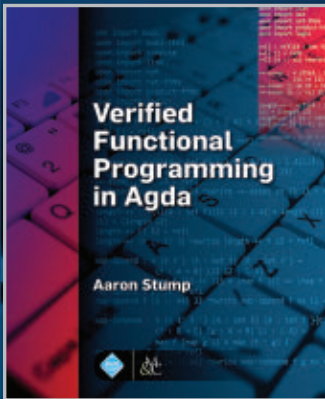
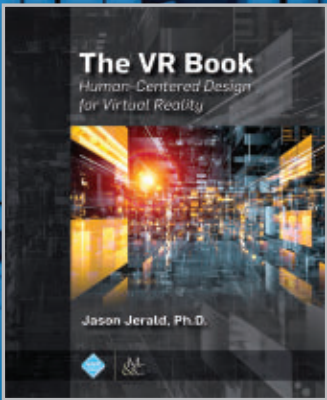
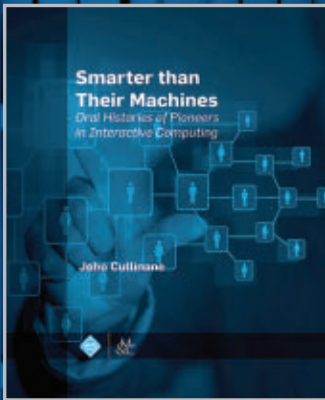
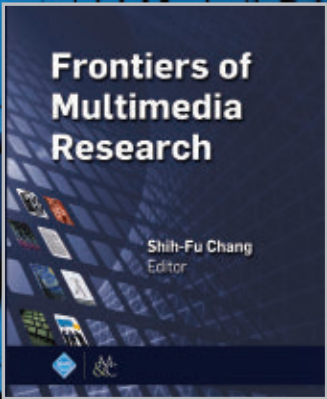
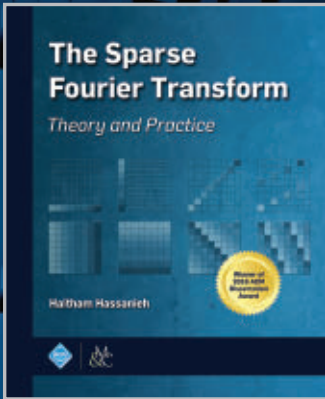
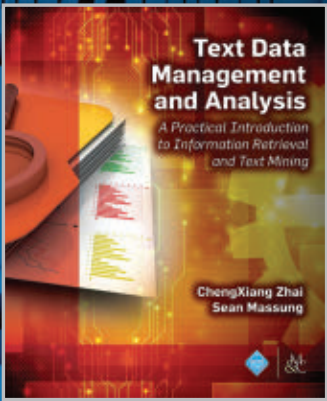
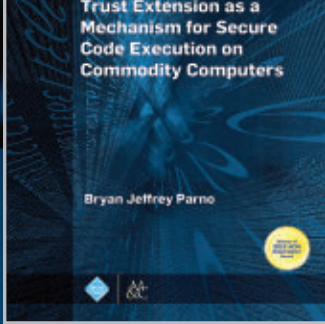
Responsible Vulnerability Disclosure in Cryptocurrencies



What Do Agile,
Lean, and ITIL
Mean in DevOps?

In Memoriam: Fran Allen

A Decade of Social Bot Detection
Strategies for spotting bots that tamper with
political elections and other events



In-depth. Innovative. Insightful.

Inspired by the need for high-quality computer science publishing at the graduate, faculty, and professional levels, ACM Books are affordable, current, and comprehensive in scope.

**Full Collection | Title List
Now Available**

For more information, please visit
<http://books.acm.org>



Association for Computing Machinery

1601 Broadway, 10th Floor, New York, NY 10019-7434, USA

Phone: +1-212-626-0658 Email: acmbooks-info@acm.org

The Essentials of Modern Software Engineering

Free the Practices from the Method Prisons!

This text/reference is an in-depth introduction to the systematic, universal software engineering kernel known as “Essence.” This kernel was envisioned and originally created by Ivar Jacobson and his colleagues, developed by Software Engineering Method and Theory (SEMAT) and approved by The Object Management Group (OMG) as a standard in 2014. Essence is a practice-independent framework for thinking and reasoning about the practices we have and the practices we need. **It establishes a shared and standard understanding of what is at the heart of software development. Essence is agnostic to any particular methods, lifecycle independent, programming language independent, concise, scalable, extensible, and formally specified.** Essence frees the practices from their method prisons.

HIGH PRAISE FOR THE ESSENTIALS OF MODERN SOFTWARE ENGINEERING

“Essence is an important breakthrough in understanding the meaning of software engineering. It is a key contribution to the development of our discipline and I’m confident that this book will demonstrate the value of Essence to a wider audience. It too is an idea whose time has come.” – Ian Somerville, St. Andrews University, Scotland (author of *Software Engineering, 10th Edition*, Pearson)

“What you hold in your hands (or on your computer or tablet if you are so inclined) represents the deep thinking and broad experience of the authors, information you’ll find approachable, understandable, and, most importantly, actionable.”
– Grady Booch, IBM Fellow, ACM Fellow, IEEE Fellow, BCS Ada Lovelace Award, and IEEE Computer Pioneer

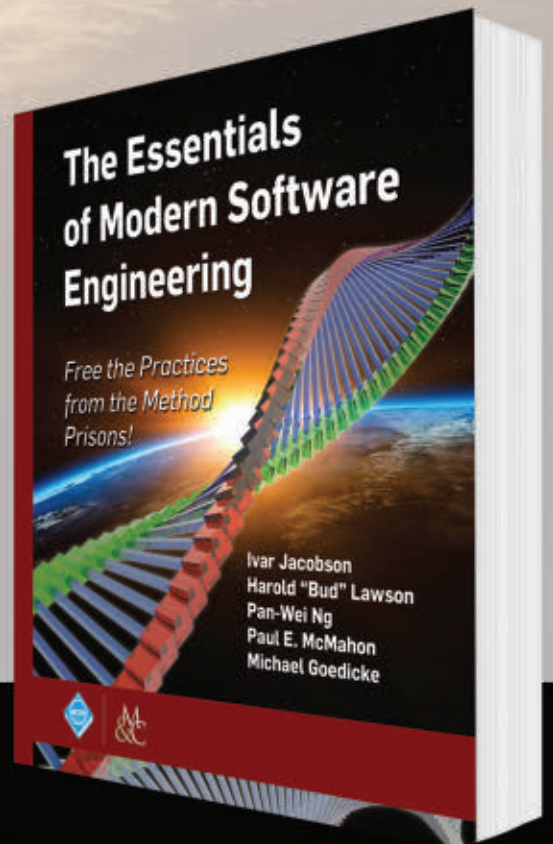
**Ivar Jacobson, Harold “Bud” Lawson,
Pan-Wei Ng, Paul E. McMahon,
Michael Goedicke**

ISBN: 978-1-947487-24-6

DOI: 10.1145/3277669

<http://books.acm.org>

<http://store.morganclaypool.com/acm>



ACM BOOKS
Collection I

Departments

- 5 **Cerf's Up**
On Digital Diplomacy
By Vinton G. Cerf
-
- 8 **BLOG@CACM**
Protecting Computers and People From Viruses
Robin K. Hill considers why the comparison of organic viruses and computer viruses is so compelling.
-
- 106 **Careers**

Last Byte

- 108 **Upstart Puzzles**
Privacy-Preserving Polling
Can you answer a poll without revealing your true preferences and have the results of the poll still be accurate?
By Dennis Shasha

News



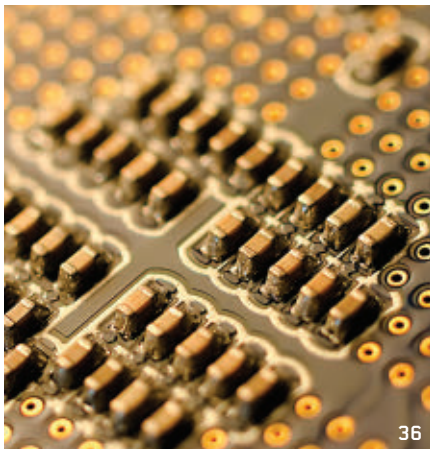
- 10 **Bouncing Balls and Quantum Computing**
A lighthearted method for calculating π is analogous to a fundamental algorithm for quantum computing.
By Don Monroe
-
- 13 **Thwarting Side-Channel Attacks**
Deep learning challenges chip security.
By Chris Edwards
-
- 15 **Who Has Access to Your Smartphone Data?**
ISPs, app developers, and even the government may know more about you than you think.
By Keith Kirkpatrick
-
- 18 **Fran Allen: 1932–2020**
By Simson Garfinkel and Eugene H. Spafford

Viewpoints

- 20 **Technology Strategy and Management**
Self-Driving Vehicle Technology: Progress and Promises
Seeking the answer to the elusive question, 'Are we there yet?'
By Michael A. Cusumano
-
- 23 **Inside Risks**
A Holistic View of Future Risks
Almost everything is somehow interrelated with everything else—and that should not surprise us.
By Peter G. Neumann
-
- 28 **Kode Vicious**
Sanity vs. Invisible Markings
Tabs vs. spaces
By George V. Neville-Neil
-
- 30 **Viewpoint**
We Need to Automate the Declaration of Conflicts of Interest
Leveraging existing data sources to improve the declaration and management of authorship conflicts of interest.
By Richard T. Snodgrass and Marianne Winslett
-
- 33 **Viewpoint**
Using Computer Programs and Search Problems for Teaching Theory of Computation
Recognizing the significance of a cornerstone of computer science.
By John MacCormick



Practice



36

36 **The History, Status, and Future of FPGAs**

Hitting a nerve with field-programmable gate arrays.

By *Oskar Mencer, Dennis Allison, Elad Blatt, Mark Cummings, Michael J. Flynn, Jerry Harris, Carl Hewitt, Quinn Jacobson, Maysam Lavasani, Mohsen Moazami, Hal Murray, Masoud Nikraves, Andreas Nowatzky, Mark Shand, and Shahram Shirazi*

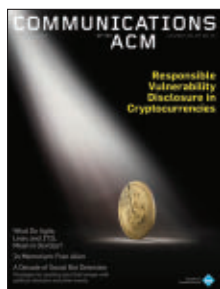
40 **Debugging Incidents in Google's Distributed Systems**

How experts debug production issues in complex distributed systems.

By *Charisma Chan and Beth Cooper*

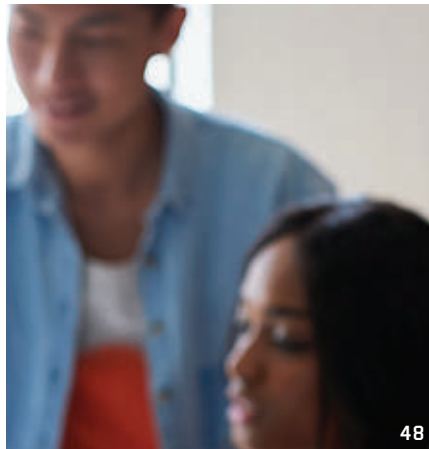


Articles' development led by [acmqueue.queue.acm.org](https://queue.acm.org)



About the Cover: Cryptocurrencies are buggy, with discoveries often uncovered at different points in the transaction process. Who's responsible for reporting those cracks and weaknesses? This month's cover story provides some answers and strategies. Cover illustration by The Image Foundation.

Contributed Articles



48

48 **What Do Agile, Lean, and ITIL Mean to DevOps?**

The value of learning skillsets within a trio of disciplines and the role each plays in DevOps.

By *Stuart Galup, Ronald Dattero, and Jing Quan*



Watch the authors discuss this work in an exclusive *Communications* video. <https://cacm.acm.org/videos/agile-lean-and-itol>

54 **Real Time Spent on Real Time**

The story of the development of a sound, static method for worst-case execution-time analysis.

By *Reinhard Wilhelm*

Review Articles

62 **Responsible Vulnerability Disclosure in Cryptocurrencies**
Software weaknesses in cryptocurrencies create unique challenges in responsible revelations.

By *Rainer Böhme, Lisa Eckey, Tyler Moore, Neha Narula, Tim Ruffing, and Aviv Zohar*



Watch the authors discuss this work in an exclusive *Communications* video. <https://cacm.acm.org/videos/vulnerability-disclosure>

72 **A Decade of Social Bot Detection**

Bots increasingly tamper with political elections and economic discussions. Tracing trends in detection strategies and key suggestions on how to win the fight.

By *Stefano Cresci*

Research Highlights

86 **Technical Perspective**
Analyzing Smart Contracts with MadMax

By *Benjamin Livshits*

87 **MadMax: Analyzing the Out-of-Gas World of Smart Contracts**

By *Neville Grech, Michael Kong, Anton Jurisevic, Lexi Brent, Bernhard Scholz, and Yannis Smaragdakis*

96 **Technical Perspective**
Two for the Price of One

By *Paul Beame*

97 **Lower Bounds for External Memory Integer Sorting via Network Coding**

By *Alireza Farhadi, Mohammad Taghi Hajiaghayi, Kasper Green Larsen, and Elaine Shi*



Vinton G. Cerf

DOI:10.1145/3418557

On Digital Diplomacy

THE TERM *DIGITAL DIPLOMACY*^a might have two interpretations. One is the conduct of diplomacy through digital means. The other is diplomacy concerning digital technologies. Both interpretations are addressed in this column. Interestingly, the term *protocol* in the computer communication sense was adopted from its use in diplomatic terms. Protocols are practices, procedures, and forms in the diplomatic world. In the world of computing, they are procedures and formats for information exchange. The conduct of diplomacy through digital means is, at first glance, a natural extension of face-to-face (F2F) and written diplomacy. From 1845 to 1865, telegraphy was a terrestrial service and quickly put to use in commerce, diplomacy, and war.^b When the first, lasting telegraphic trans-Atlantic cable was laid in 1866,^c telegraphy became an important part of rapid diplomacy on an intercontinental scale. The invention of the telephone provided for real-time interactions, the value of which became especially apparent in the wake of the October 1962 “Cuban Missile Crisis” leading to the installation of a “hotline” between the White House and the Kremlin. Facsimile transmission capability arrived in quantity in the 1980s, transforming diplomacy once again.

With the arrival of the Internet, electronic messaging has become widespread and in informal use worldwide. Formal diplomatic communications

are still largely conducted by fax, post (including overnight delivery), and telephone as well as venerable F2F communication. Electronic mail is typically used for internal but not external communication owing to the potential for misrepresenting the origin of public email messages (“spoofing”). Even that is slowly changing as various forms of strong authentication become available.

The arrival of the World Wide Web (WWW) in 1991 heralded a new era in information production and sharing. From the diplomatic perspective, it offered an open source for intelligence about events in the world and also an avenue to engage in what some call *soft diplomacy* by which is meant pursuit of diplomatic aims by means of suasion and public pressure. Concurrent with the evolution of the WWW, cloud computing has emerged as the natural successor to the time-sharing systems of the 1960s. Cloud computing and smartphones have contributed to two more recent dramatic changes in the online environment that are upending traditional diplomatic communication. The first is the arrival of social media, most notably Facebook, Twitter, Instagram, Tik-Tok, WeChat, and YouTube, among others. The second is public and private videoconferencing. Videoconferencing is not new. Indeed, its origins lie in the 1960s and experiments with this medium were conducted on the American ARPANET in the 1970s. The COVID-19 pandemic, however, has launched videoconferencing over the Internet into orbit. To account for lockdowns and social distancing, videoconferencing systems like Zoom, Teams, Meet, among others, are now in daily use worldwide. We are locked in front of our laptop screens, not only by email, chat, and surfing the Web, but by videoconferences and webinars going day and night.

On the negative side of the ledger, social media have become avenues for the spread of misinformation and disinformation and have been used to foment civil unrest. The latter is often triggered by the distribution of false information and reinforced by the use of botnets^d and fake accounts in social media. Indeed, such abuses highlight the growing recognition in government and diplomatic circles that abuse of the Internet is a national and international problem that needs attention (See S. Cresci’s article on p. 72). Hence the second interpretation of *digital diplomacy*: diplomatic negotiations about dealing with the abuse of digital infrastructure. The canvas for this debate is enormous. There are billions of digital devices in use today from mobile smartphones, laptops, desktops, tablets, and a growing body of devices called collectively the Internet of Things. Hacking of these devices along with the distribution of malware (that is, harmful software) constitute major hazards in the online world. These hazards are made all the more difficult to deal with because the Internet is global, and perpetrators may be in one jurisdiction while the victim is in another.

It is apparent that the world needs thoughtful and technically credible debate on alternatives for containing the problem of harmful behavior on the Internet. Perpetrators must be identified, and international norms and agreements established for bringing them to justice. This is a task for diplomacy about digital technology and its use and abuse, and computer scientists have a serious role to play in this discussion. ■

^d A “bot” is a computer that has been hacked to become part of a large network of machines that can be used to mount denial of service attacks, spread spam and misinformation and reinforce tweet storms, among other things. A “botnet” is a collection of bots.

Vinton G. Cerf is vice president and Chief Internet Evangelist at Google. He served as ACM president from 2012–2014.

Copyright held by author.

^a O.S. Adesina, J. Summers (Reviewing Ed.) Foreign policy in an era of digital diplomacy. *Cogent Social Sciences* 3, 1 (2017). DOI: 10.1080/23311886.2017.1297175

^b President Abraham Lincoln famously and avidly read dispatches from the war front during the Civil War, for example.

^c Earlier efforts beginning in 1854 did not succeed or failed quickly.

ACM ON A MISSION TO SOLVE TOMORROW.



Dear Colleague,

Without computing professionals like you, the world might not know the modern operating system, digital cryptography, or smartphone technology to name an obvious few.

For over 70 years, ACM has helped computing professionals be their most creative, connect to peers, and see what's next, and inspired them to advance the profession and make a positive impact.

We believe in constantly redefining what computing can and should do.

ACM offers the resources, access and tools to invent the future. No one has a larger global network of professional peers. No one has more exclusive content. No one presents more forward-looking events. Or confers more prestigious awards. Or provides a more comprehensive learning center.

Here are just some of the ways ACM Membership will support your professional growth and keep you informed of emerging trends and technologies:

- Subscription to ACM's flagship publication *Communications of the ACM*
- Online books, courses, and videos through the **ACM Learning Center**
- Discounts on registration fees to ACM Special Interest Group conferences
- Subscription savings on specialty magazines and research journals
- The opportunity to subscribe to the **ACM Digital Library**, the world's largest and most respected computing resource

Joining ACM means you dare to be the best computing professional you can be. It means you believe in advancing the computing profession as a force for good. And it means joining your peers in your commitment to solving tomorrow's challenges.

Sincerely,

A handwritten signature in black ink, appearing to read 'G. Kotsis'.

Gabriele Kotsis
President
Association for Computing Machinery



Association for
Computing Machinery

Advancing Computing as a Science & Profession

SHAPE THE FUTURE OF COMPUTING. JOIN ACM TODAY.

www.acm.org/join/CAPP

SELECT ONE MEMBERSHIP OPTION

ACM PROFESSIONAL MEMBERSHIP:

- Professional Membership: \$99 USD
- Professional Membership plus ACM Digital Library: \$198 USD (\$99 dues + \$99 DL)

ACM STUDENT MEMBERSHIP:

- Student Membership: \$19 USD
- Student Membership plus ACM Digital Library: \$42 USD
- Student Membership plus Print *CACM* Magazine: \$42 USD
- Student Membership with ACM Digital Library plus Print *CACM* Magazine: \$62 USD

- Join ACM-W:** ACM-W supports, celebrates, and advocates internationally for the full engagement of women in computing. Membership in ACM-W is open to all ACM members and is free of charge.

PAYMENT INFORMATION

Name

Mailing Address

City/State/Province

ZIP/Postal Code/Country

- Please do not release my postal address to third parties

Email Address

- Yes, please send me ACM Announcements via email
- No, please do not send me ACM Announcements via email

- AMEX VISA/MasterCard Check/money order

Credit Card #

Exp. Date

Signature

Purposes of ACM

ACM is dedicated to:

- 1) Advancing the art, science, engineering, and application of information technology
- 2) Fostering the open interchange of information to serve both professionals and the public
- 3) Promoting the highest professional and ethics standards

By joining ACM, I agree to abide by ACM's Code of Ethics (www.acm.org/code-of-ethics) and ACM's Policy Against Harassment (www.acm.org/about-acm/policy-against-harassment).

I acknowledge ACM's Policy Against Harassment and agree that behavior such as the following will constitute grounds for actions against me:

- Abusive action directed at an individual, such as threats, intimidation, or bullying
- Racism, homophobia, or other behavior that discriminates against a group or class of people
- Sexual harassment of any kind, such as unwelcome sexual advances or words/actions of a sexual nature

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.



ACM General Post Office
P.O. Box 30777
New York, NY 10087-0777

1-800-342-6626 (US & Canada)
1-212-626-0500 (Global)
Hours: 8:30AM - 4:30PM (US EST)

Fax: 212-944-1318
acmhelp@acm.org
www.acm.org/join/CAPP

The *Communications* Web site, <http://cacm.acm.org>, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.



Follow us on Twitter at <http://twitter.com/blogCACM>

DOI:10.1145/3415748

<http://cacm.acm.org/blogs/blog-cacm>

Protecting Computers and People From Viruses

Robin K. Hill considers why the comparison of organic viruses and computer viruses is so compelling.



Robin K. Hill
The Virus Analogy and Validation
<https://bit.ly/2yTFYCX>
May 29, 2020

The COVID-19 pandemic highlights the virus analogy that gave rise to the use of the word “virus” from biology, to label a malicious program that attacks computer systems. The situation moves us to look into that, as another way to compare nature and artifact, and as an excuse to raise more abstract questions. We are moved also to stipulate that our mastery of both the biological and computational forms is shallow, and to invite other, better observations to follow. See Aprville and Guillaume¹ for greater depth and intriguing crossover speculation, Weis¹¹ for yet more intriguing comparison, and Wenliang Du’s website for detailed virus examples,³ which constitute dramatic reading for coders.

A virus is generally not regarded as a living organism, but sometimes described as (similar to) software. When

the first self-replicating computer programs made the rounds, they were experiments or pranks;¹² for most, the point was solely reproduction. An early computer worm was beneficent, but escaped control.²

We distinguish computer viruses from computer worms by the profligate scale of replication, viruses generating a broadcast of copies rather than a chain of copies. The obvious points of analogy across both types of virus include that viruses are tiny, invading a host much greater in size and complexity, without an overt signal, and that viruses disrupt some process in the host. Neither computer nor biological virus *necessarily* does damage. In biology, self-replication is an end, not a means, making the damage a side-effect. In the modern computer virus, the end is likely to be the action of a payload of malicious code. Now the term “virus,” in both environments, connotes an intrusive and damaging force carrying dangerous baggage.

To explore some points of analogy systematically, consider *access*: How is virus entry accomplished? Computer

viruses look for an opening by probing known vulnerabilities; if one is found, malicious code is injected. This is quite like the organic version.

Consider *gain*: What does the virus get out of this, and how? The virus gets more virus, and the means of reproduction is the same—self-replication. Note the correspondence to the Unix system `fork()` call, which spawns a new process by replicating the current process. The history tells us that this happened because it was easy: “... it seems reasonable to suppose that it exists in Unix mainly because of the ease with which *fork* could be implemented without changing much else.”⁸ The heuristic, across both types: To start a new working structure, just copy the working structure on hand.

Consider *pathology*, the means of damage. A virus damages the host body by depleting cell resources, consumed by the virus; bursting the cell walls; or generating toxic byproducts. Do each of these have an analogy? Sure—Denial of Service; breaching buffer boundaries or reverse shell; interference with

the operating system, degrading its protection of system resources such as CPU cycles, files, and ports.^{2,10}

We could consider *defense*, the host's prevention or cure mechanism, that is, the action taken if the host somehow notices that something is wrong. That panoply of fascinating mechanisms is beyond our expertise, but it is clear vaccination is one of them, leading to countermeasures such as mutation. Both organic and computer viruses can mutate quickly. But mutation in organics is a quirk, a random unguided alteration. Mutation in computer programs is human-directed. The brute-force options for repair and defense are off limits to humans. We can't reboot to reset memory, let alone re-install a clean operating system.

Viruses have been described as troops in a war game, initiating and reacting, as they take over cells for the purpose of replication. But wait—Is there a *purpose*? All we can say for sure is that viruses insert genetic material into cells, which causes the cells to generate more viruses. Is there a struggle? Is control being deliberately wrested from the cell, or is there actually no agent involved that gives a hoot, no intention at all? The vocabulary of aggression in cell science (layperson's version) reflects our human phenomenology, projected onto what we see. It may be fair, or it may be distorted. It may be way off the mark; the cells might be “fulfilled”—an odd thought. But why is it less odd to say the cells are “defeated”? Why use the language of attack, when the language of hospitality (or indifference) might model the process just as well (the language of indifference, even better)? Previously, we said that in biology, damage is a “side-effect,” which assumes some kind of intention. We now question that assumption. Other natural forces bring about change; the wind threatens, intrudes, and damages, but to speak of its intention is only poetic.

In computing, similarly, a computer virus executes in order to create more copies of its code and then disseminate them. Does that statement of the analogy, through the phrase “in order to,” lead us to the attribution of volition to the computer virus, inaccurately? We claim it is misleading to speak as if the organic virus has volition. Imbued by the programmer, however, a computer virus ex-

“The really interesting question is what a strong successful analogy, matching computer viruses to organic viruses, would mean.”

hibits hostility. But wait. That means the computer virus is more like the organic virus than the organic virus itself!

Of course, the question of volition, seen here on a small scale, bears on larger questions in the philosophy of computing as well, those in artificial intelligence and cognitive science connected to intentionality and consciousness. That inquiry could be aided by a new locution for computer virus, which might even inform a new locution for organic virus. My earlier article “Articulation of Responsibility”⁷ called for such locutions.

Programs do not make decisions. Because it looks like they do, we need a way to talk about what they actually do that is not misleading. Viruses do not “intend” in any meaningful way; they just behave as if they were intending. Or perhaps they don't even “behave” in any particular way, they just exhibit actions that intentional beings would exhibit if they had as a goal the end-result reached by the virus. We are so dependent on the vocabulary of intention and volition that we have no other non-awkward options.

Analogies between natural and computation phenomena, tight or stretched, have formed the subjects of several pieces in this space.⁴⁻⁶ In the case before our eyes, we see the analogy between the biological virus and the computer virus exhibits strengths and weaknesses, and may offer further possibilities. Points of positive similarity may not be due to cause and effect, but rather to effects of some common cause, something like the general vulnerability of processes that use input

and output. We might even propose the proper analogue to the biological microbe is the programmer-code pair, a self-contained system that lies between the program and the programmer, enjoying some kind of collective semi-animate agency. We can turn to philosophy to ask—Do agents have to be individual and human? That's debatable⁹ beyond the scope of this inquiry.

But wait. The really interesting question is what a strong successful analogy, matching computer viruses to organic viruses, would mean. Does it mean that some common notions—say, the general vulnerability of input (as mentioned previously), or entry through a defined interface, or subversion of a external body's resources—are somehow universal? If so, have we gained anything beyond a pleasant self-validation? But wait! What does validation get us, anyway? Are computer scientists to congratulate ourselves when our artifacts look like nature? What's so great about that? Or *is* there something great about that? If so, what's not so great about artifice? **■**

References

1. Aprville, A. and Lovet, G., 2012. *An Attacker's Day into Human Virology*. Appendix comprises a table of vocabulary analogs.
2. Chen, T. and Robert, J.M., 2004. The evolution of viruses and worms. In Chen, W. (Ed.). (2004). *Statistical Methods in Computer Security*. Boca Raton: CRC Press.
3. Du, W., Undated. Computer & Internet Security: Videos, Slides, Problems and Labs. Website for the book *Computer & Internet Security: A Hands-on Approach*, Second Edition. 2019.
4. Hill, R.K., 2016. Fiction as Model Theory, BLOG@CACM, December 30, 2016.
5. Hill, R.K., 2017. Operating Systems as Possible Worlds. BLOG@CACM, April 29, 2017.
6. Hill, R.K., 2017. Human Acts and Computer Apps. BLOG@CACM, November 28, 2017.
7. Hill, R.K., 2018. Articulation of Decision Responsibility. BLOG@CACM, May 21, 2018.
8. Ritchie, D.M. 1980. The evolution of the Unix time-sharing system. In *Proceedings of the Symposium on Language Design and Programming Methodology*. Springer, 1980. <https://bit.ly/3i5Zo8w>
9. Schlosser, M. Agency. *The Stanford Encyclopedia of Philosophy*, Winter 2019 Edition. E.N. Zalta, Ed.
10. Various experts. When and how did the metaphor of the computer 'virus' arise? *Scientific American*, <https://bit.ly/31nebos>. Article lists answers to the given question, usually identifying Fred Cohen, student of Adleman at University of Southern California. September 2, 1997.
11. Weis, O. What if it was a software bug/virus? Cyber vs. COVID-19: A thought experiment. Rookout, <https://bit.ly/2ERhi00>
12. Wikipedia contributors. Creeper (program). In Wikipedia, *The Free Encyclopedia*. May 29, 2020.

Robin K. Hill is a lecturer in the Department of Computer Science and an affiliate of both the Department of Philosophy and Religious Studies and the Wyoming Institute for Humanities Research at the University of Wyoming. She has been a member of ACM since 1978.

Bouncing Balls and Quantum Computing

A lighthearted method for calculating π is analogous to a fundamental algorithm for quantum computing.

THE HISTORY OF science and mathematics includes many examples of surprising parallels between seemingly unrelated fields. Sometimes these similarities drive both fields forward in profound ways, although often they are just amusing.

In December, Adam Brown, a physicist at Google, described a surprisingly precise relationship between a foundational quantum-computing algorithm and a whimsical method of calculating the irrational number π . “It’s just a curiosity at the moment,” but “the aspiration might be that if you find new ways to think about things, that people will use that to later make connections that they’d not previously been able to make,” Brown said. “It’s very useful to have more than one way to think about a given phenomenon.”

In a preprint posted online (but not yet peer-reviewed at press time), Brown showed a mathematical correspondence between two seemingly unconnected problems. One is the well-known Grover search algorithm proposed for quantum computers, which should be faster than any classical equivalent. The other is a surprising



procedure in which counting the number of collisions between idealized billiard balls produces an arbitrarily precise value for the π .

Quantum Algorithms

Quantum computing exploits quantum bits, or qubits, such as ions or superconducting circuits, that can simultaneously represent two distinct states. In principle, a modest number

of qubits can represent and manipulate an exponentially larger number of combinations. Exploiting this possibility for computing seemed like a pipe dream, however, until researchers devised algorithms to extract useful information from the qubits. The first such algorithm, described in 1994 by Peter Shor, then at Bell Labs in New Jersey, efficiently finds the prime factors of a number, poten-

tially cracking important cryptography schemes. The trick is to frame the problem as determining the repetition period of a sequence, essentially a Fourier transform, which can be found using global operations on an entire set of qubits.

The second fundamental algorithm, devised in 1996 by Lov Grover working independently at Bell Labs, operates quite differently. “Shor and Grover are the two most canonical quantum algorithms,” according to Scott Aaronson of the University of Texas at Austin. “Even today, the vast majority of quantum algorithms that we know are recognizably either ‘Shor-inspired’ or ‘Grover-inspired’, or both.”

Grover’s algorithm is often described as a database search, examining a list of N items to find the item that has a desired property. If the list is ordered by some label (for example, alphabetized), any label can be found by repeatedly dividing the list in successive halves, eventually requiring $\log_2 N$ queries. For an unsorted list, however, checking each item in turn requires, on average $N/2$ steps (and possibly as many as N).

Like other quantum algorithms, Grover’s manipulates the entire set of qubits simultaneously, while preserving the relationships between them (prematurely querying any qubit to determine its state turns it into an ordinary bit, squandering any quantum advantage). However, Grover showed the desired item can generally be found with only $\frac{\pi}{4}\sqrt{N}$ global operations.

This improvement is less than that seen in Shor-style algorithms, which typically are exponentially faster than their classical counterparts. The Grover approach, however, can be applied to more general, unstructured problems, Brown notes.

The calculation starts with an equal admixture of all N qubits. The algorithm then repeatedly subjects all the qubits to two alternating manipulations. The first operation embodies the target: it inverts the state of a specific, but unknown, bit. The task is to determine which bit is altered, but not by measuring them all. The second operation does not require any information about the target. Grover

Grover’s algorithm manipulates the entire set of qubits simultaneously, while preserving the relationships between them.

found that each time this sequence is repeated, the weight of the target in the admixture increases (although this cannot be measured). After the correct number of repetitions, there is an extremely high chance a measurement will yield the correct result.

Bouncing Billiards

These sophisticated quantum manipulations may seem to have little relationship to bouncing billiard balls. Yet Brown, while working on issues related to Grover’s algorithm, came across an animation by math popularizer Grant Sanderson that made him notice the similarities. In his paper, Brown shows there is a precise mapping between the two problems.

Sanderson’s animation illustrates a surprising observation described in 2003 by Gregory Galperin, a mathematician at Eastern Illinois University in Charleston. In the paper “Playing Pool with π ,” he imagined two billiard balls moving without friction along a horizontal surface, bouncing off each other and off a wall on the left side in completely elastic collisions (which preserve their combined kinetic energy).

If the right-hand ball is sent leftward toward a second stationary ball that is much lighter, the smaller ball will be sent back toward the left-hand wall without slowing the larger ball much. The small ball will bounce off the wall, and then collide with the large one again, repeating this multiple times. Eventually the collisions will turn the large ball around until it finally escapes to the right faster than the small ball can pursue it.

The number of collisions needed before this escape can occur grows

larger with the ratio of the mass of the large ball compared to the small one. If the masses are equal, it will take three bounces: the first transfers all motion from the right ball to the left one, which bounces off the wall and then transfers its momentum back to the right ball again. If the large ball is 100 times as massive, the process will take 31 bounces. If the mass ratio is 10,000, there will be 314 bounces. In a spectacularly impractical computation, for every increase of a factor of 100 in the mass ratio, the number of collisions (divided by the square root the mass ratio) includes another digit to the digital representation of π , 3.141592654...

Brown fortuitously encountered Sanderson’s animation (which uses blocks instead of balls) when Grover’s algorithm was fresh in his mind, and recognized significant similarities between the two situations. The two quantum operations, for example, correspond respectively to collisions between the balls and between the lighter ball and the wall. The mass ratio corresponds to the size of the database. Moreover, the final result was that the number of operations (or bounces) is proportional to π and to the square root of this size or mass ratio. (There are also two factors of two that reflect simple bookkeeping differences between the problems.)

Beyond the surprising connection between such different systems, what on earth is the number π doing in both cases? This irrational number is of course best known as the ratio of the circumference of a circle to its diameter, although it also appears in the proportions of ellipses, as well as higher-dimensional objects like spheres. One way to define a circle is through an algebraic constraint on the horizontal and vertical coordinates, x and y : The points of a circle with radius r are constrained to satisfy $x^2 + y^2 = r^2$.

As it turns out, both the billiard problem and the Grover algorithm have constraints of this form. Collisions of the balls or manipulations of the quantum system correspond to rotations along the circle defined by these constraints.

For example, for two billiards of mass m (with velocity v_m) and M (with velocity v_M), an elastic collision preserves their total kinetic energy,

$\frac{1}{2}mv_m^2 + \frac{1}{2}Mv_M^2$. Completely reversing the velocity of the larger ball requires a total “rotation” by 180° (π radians) in the plane with coordinates v_m and v_M .

Similarly, for quantum systems, the probability of observing a particular outcome is proportional to the square of the “wave function” corresponding to that outcome. The sum of the probability (squared amplitude) for the target and all other outcomes must be one.

Historical Examples of Connections

There is still the question, “Is this profound insight into the nature of reality, or is it just a sort of curiosity?” Brown said. “Maybe Grover search is telling us something profound about the nature of reality, and maybe the bouncing-ball thing is more of a curiosity, and maybe connecting them is more in the spirit of the second one than the first one.”

Still, there have been numerous cases in physics, and especially in mathematics, where such connections have contributed profoundly to progress. For example, physicists have spent more than two decades exploring a surprising correspondence between strongly interacting multi-

For quantum systems, the probability of seeing a particular outcome is proportional to the square of the wave function corresponding to that outcome.

particle quantum systems and gravitational models incorporating curved spacetime with one higher dimension. There is even hope the wormholes in spacetime can help resolve paradoxes associated with quantum-mechanical “entanglement” of distant particles.

Mathematics has frequently advanced through connections between disparate fields. For example, Fermat’s “last theorem,” involving integer solutions of a simple equation, was only proved centuries later using

methods from “elliptic curves.” In another example, in January, computer scientists proved a theorem relating entanglement to Alan Turing’s notion of decidable computations, which continues to shake up other seemingly unrelated fields.

For his part, Aaronson suspects the Grover-billiard correspondence, although “striking in its precision,” is probably “just a cute metaphor (in the sense that I don’t know how to use it to deduce anything about Grover’s algorithm that we didn’t already know). And that’s fine.” □

Further Reading

Galperin, G.,

“Playing Pool with π : The Number π from a Billiard Point of View,” *Regular and Chaotic Dynamics* 8, p. 375 (2003).

Brown, A.R.,

“Playing Pool with $|\psi\rangle$: from Bouncing Billiards to Quantum Search,” *arXiv.org:1912.02207* (2019).

Sanderson, G.,

“How Pi Connects Colliding Blocks to a Quantum Search Algorithm,” *Quanta* (2020)

Don Monroe is a science and technology writer based in Boston, MA, USA.

© 2020 ACM 0001-0782/20/10 \$15.00

ICCCQ

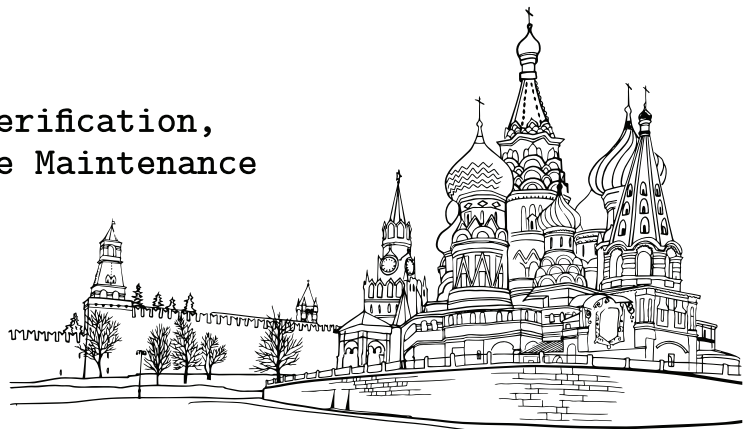
www.iccq.ru

The First International Conference on Code Quality
in cooperation with IEEE Computer Society

Moscow, Russia
27 Mar 2021

Static Analysis, Program Verification,
Bug Detection, and Software Maintenance

CfP closes:
4 Dec 2020



Thwarting Side-Channel Attacks

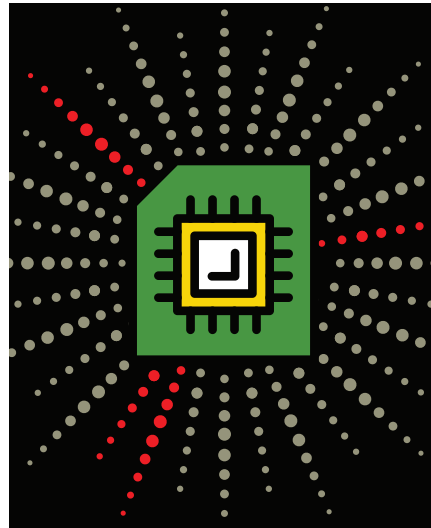
Deep learning challenges chip security.

THE SAME ATTRIBUTES that give deep learning its ability to tell images apart are helping attackers break into the cryptoprocessors built into integrated circuits that were meant improve their security. The same technology may provide the tools that will let chip designers find effective countermeasures, but it faces an uphill struggle.

Side-channel attacks have been a concern for decades, as they have been used in the hacking of smartcard-based payment systems and pay-TV decoders, as well as in espionage. Yet the rise of Internet of Things (IoT) and edge systems and their use in large-scale, commercially sensitive applications makes such attacks a growing worry for chipmakers. The innate connectivity of IoT devices means success in obtaining private encryption keys from them may open up network access on cloud-based systems that rely on their data.

Although there are side-channel attacks that can be deployed remotely by measuring the timing of responses from software running on a server, many of the most pernicious attacks rely on physical proximity and can be performed using low-cost electrical instruments (see “Secure-System Designers Strive to Stem Data Leaks,” *Communications*, April 2015). The switching of logic gates creates changes in the electromagnetic fields around them that can be detected by probes placed close to the chip’s surface by an attacker. Another side channel stems from rapid changes in energy consumption that can be seen by attaching probes to the device’s power-supply connections.

Though it is not the only approach, a common technique is to profile the target using known keys and plain text, collecting thousands of traces of the emissions of the process. Careful analysis of the traces often will show cor-



relations between power or emissions spikes, and the value of the byte of the encryption key being processed during that interval.

Alric Althoff, principal engineer at secure-hardware tools supplier Tortuga Logic, says, “The majority of the attacks involve very straightforward statistics.”

Often, the signals that show a dependency on data are restricted to a few samples within the traces, which may cover thousands of samples that are little more than noise. One approach the design community has tried to apply is to use formal models of computation to predict how much data-dependency is present in the switching of logic gates, and using that information to try to remove correlated samples from appearing in the traces.

“Design based on the theoretical models does mitigate the vast majority of the leakage. The problem is that, if you are free to collect millions of traces and analyze them, correlations are going to start popping out even if they are very small,” Althoff says. A second issue is that the layout of transistors on the silicon die lead often leads to larger emissions than predicted. “Those are

not modeled using formal techniques,” he adds, though improvements in design tools may lead to simulators that can perform accurate-enough assessment without having to go to actual silicon before testing.

The elliptic-curve code known as Curve25519 benefited from algorithmic modeling so it could more easily resist side-channel analysis. However, in 2017, a team from the University of Pennsylvania found smartphone circuitry that clearly leaked the positions of zeroes in intermediate calculations that pointed directly to subkey bytes.

Many countermeasures attempt to hide the samples that correlate well with key data. One common approach is masking, which combines the actual key bytes with randomly chosen dummy values just before the most vulnerable part of the encryption sequence that is most often targeted by adversaries: often where the key is first used to disguise the plaintext data. Dummy operations make it more difficult for hackers to align traces and find the correlations embedded in them.

At the International Solid State Circuits Conference (ISSCC) in February, researchers from Purdue University and the Georgia Institute of Technology showed how the bulk of electromagnetic emissions come from the layers of metal interconnect closest to the top of an integrated circuit. Disconnecting a cryptoprocessor from direct access to the three highest metal layers, the team cut usable interference significantly in one experiment.

A second project led by the Purdue team, presented at the Custom Integrated Circuits Conference (CICC) a month later, isolated the cryptoprocessor behind an on-chip power regulator that prevents an external observer from seeing the small changes in current that reveal circuit behavior. At the same conference, Intel Labs’ director of circuit

technology research Vivek De described a number of methods that use on-chip power converters to inject noise into the power signals that an attacker would typically probe at the PCB level. He claimed the techniques could cut by five orders of magnitude the signal-to-noise ratio.

Althoff says, “The correlation with the signal-to-noise is very well understood. For a specific reduction in correlation, you want a certain amount of noise, and you can compute that.”

The growing question is determining how much noise you need to inject to maintain secrecy. Attackers are moving from conventional statistical tools to deep learning because it readily combines data from disparate positions across traces, rather than focusing attention on a small number of what they hope are telltale samples. In doing so, deep learning reverses the effects of countermeasures. The convolutional filters often used in DNNs to detect features no matter where they are in images appear to be effective at filtering out the noise introduced by masks and dummy operations.

However, deep learning is far from a reliable tool. Althoff points out that subtle artifacts in trace captures, such as a constant offset in the underlying amplitude, do not cause problems for statistical models, but can easily throw off a machine-learning pipeline. Research has shown if training is not carefully controlled, deep-learning models are highly prone to overfitting, which significantly reduces their ability to correctly predict keys when tested on traces they have not seen before.

Guilherme Perin, senior security analyst at Riscure, says another technique that tends to show better performance overall is ensemble learning using subtly different neural-network models and fed with complementary subsets of the traces obtained during profiling.

Because deep learning can be highly unreliable when used in attacks, its use poses bigger problems for those building defenses who try to apply it as a form of penetration testing.

“If you attack something successfully, you’ve proven that it’s vulnerable. If you’re not successful, you’ve proven nothing,” Althoff notes. “The best performing attacks in nature are unpublished. Attackers doing nefarious

“There is a trend in explainable learning, to really understand why the models are making the decisions they are making.”

things are not sharing their approaches. An attacker in the business of attacking is going to run many at once and will have a whole set of scripts set up to help perform them.”

There is some hope that machine learning will provide an answer as to whether designs are vulnerable, and this is one reason why Riscure is pursuing it. However, rather than training models just to find keys, the Dutch consultancy’s approach focuses on looking inside the model that training creates. Perin says one approach that has demonstrated some success is to focus on the neurons that have the strongest influence on correctly predicted key bytes and tracing back to the combinations of samples that activate them. Although sifting through this data is a manually intensive process today, it may provide the basis for automated systems that identify which operations in a long sequence are leaking the most information.

Althoff sees potential in analyzing the models created by techniques such as deep learning. “There is a trend in explainable learning, to really understand why the models are making the decisions they are making. We want to look at the weights that correspond to a certain region and why it weights them more heavily.”

What makes better analysis of information leakage increasingly important is the cost and energy overhead of many of the countermeasures. The Purdue team claims their on-chip regulator made it impossible for a DNN to distinguish data operations from noise on the power rails. Unfortunately, it has a power overhead of 50% when the cryptoprocessor is running. It also adds to cost through increased silicon area.

“Many mitigation techniques come with major overheads in power performance and die area that are impractical for IoT devices,” De says.

De says one option is to only invoke strong countermeasures if the device detects behavior that indicates side-channel analysis is under way. One method is direct; he points to an idea proposed five years ago by a team from Tohoku and Kobe universities in Japan; they placed inductors on the surface of the IC at strategic points to pick up the distortion in electric field caused by a nearby measurement probe. Another technique might be to monitor how the cryptoprocessor is being used, and whether that points to a large number of operations being profiled (although this is vulnerable to false positives).

The game of cat and mouse will continue until researchers develop better tools to determine how much information circuitry leaks, and what are the limits of detection. □

Further Reading

Das, D., Danial, J., Golder, A., Ghosh, S., Raychowdhury, A., and Sen, S.

Deep Learning Side-Channel Attack Resilient AES-256 using Current Domain Signature Attenuation in 65nm CMOS
Proceedings of the 2020 IEEE Custom Integrated Circuits Conference (CICC)

Edwards, C.

Secure-System Designers Strive to STEM Data Leaks, *Communications*, April 2015, 18-20, <http://bit.ly/38MvW28>

Masure, L., Dumas, C., and Prouff, E.

A Comprehensive Study of Deep Learning for Side-Channel Analysis
IACR Cryptology ePrint Archive (2019), <https://eprint.iacr.org/2019/439>

Perin, G., Ege, B., and Chmielewski, L.

Neural Network Model Assessment for Side-Channel Analysis
IACR Cryptology ePrint Archive (2019), <https://eprint.iacr.org/2019/722>

Homma, N., Hayashi, Y., Miura, N., Fujimoto, D., Tanaka, D., Nagata, M., and Aoki, T.

EM Attack is Non-Invasive? Design Methodology and Validity Verification of EM Attack Sensor, *Proceedings of the 2014 Conference on Cryptographic Hardware and Embedded Systems (CHES 2014)*, Lecture Notes in Computer Science, vol 8731.

Chris Edwards is a Surrey, U.K.-based writer who reports on electronics, IT, and synthetic biology.

Who Has Access to Your Smartphone Data?

ISPs, app developers, and even the government may know more about you than you think.

IN A WORLD that is defined by the generation and collection of data by technology and communications companies, personal information—including where people go, with whom they associate, what they purchase, and what they read, listen to, and even eat—it is quite a simple task to create a detailed profile of an individual based solely on the data captured in his or her phone.

The right to access and use the cache of personal information stored in each person's smartphone has become a major question about balancing personal privacy rights against governments' desire to monitor and retrieve data about its citizens' activities for law enforcement, public safety, and health issues. While much of the attention over the past several years has focused on demands from law enforcement to access this data to aid in criminal investigations, the COVID-19 pandemic of 2020 has refocused the debate on the government's right to access location data during health or other public safety emergencies.

Within the U.S., the primary communications privacy law that regulates the disclosure of and access to electronic data held by communication services providers, including wireless carriers, Internet Service Providers (ISPs), social media platforms, and search companies, among others, is the Electronic Communications Privacy Act of 1986 (ECPA) which, along with the Uniting and Strengthening America by Providing Appropriate Tools Required to Intercept and Obstruct Terrorism (USA PATRIOT) Act OF 2001, protects wire, oral, and electronic communications while those communications are being made, are in transit, and when they are stored on computers. As the



Act explicitly states, “Some information can be obtained from providers with a subpoena; other information requires a special court order; and still other information requires a search warrant.”

Andrew Crocker, senior staff attorney on the Electronic Frontier Foundation's civil liberties team, says the ECPA generally “requires the government to use legal process to get data about users,” rather than simply allowing them to request and receive information from service providers.

Similarly, the Fourth Amendment of the U.S. Constitution requires law enforcement agencies to demonstrate probable cause when seeking historical location data from an individual's phone. This requirement was affirmed via a 2018 Supreme Court decision, in which Chief Justice John Roberts, writing on behalf of the majority, held that police are required to obtain a warrant in ordinary investigations, but could access such information without a warrant in an emergency, such as during a bomb threat, kidnapping, or other exigent circumstance where time is of the essence. However, law enforcement

agencies do need to obtain a warrant after the imminent threat has passed, to demonstrate the inquiry was made in good faith.

“It would of course be more efficient to allow law enforcement officials to decide who to surveil on their own, without oversight by a court, but that would risk invasive surveillance at the whim of the government,” says Crocker. “If there is a true emergency that makes getting a warrant impractical, such as an imminent threat to someone's life, the Fourth Amendment and these laws allow for a brief warrantless search, often requiring the government to come back to a court after the fact.”

Former law enforcement officials agree, noting that there will always be some tension between the desire to protect personal privacy and the clear value of information that can be used to solve crimes or keep the public safe.

“It's definitely a challenge, and it's a balance between personal freedom and the ability of law enforcement to do their job, especially during the emergency circumstances that are putting others in harm's way,” says Dan-

iel Linskey, managing director of the Security Risk Management practice of New York-based risk solutions provider Kroll, and head of the company's Boston office. Linksey, a former superintendent-in-chief of the Boston Police Dept., notes that in addition to the requirement to get a warrant to obtain information, the business models of Google and other collectors of personal data generally are focused on protecting privacy, rather than making it easy for law enforcement to access such data.

"I think there's a business decision to not share information with law enforcement unless absolutely necessary," Linskey says. "And even when necessary, the resources are not in place to make that a quick or timely process. The number of requests for information has overwhelmed Google, Yahoo, and any of those data providers to keep up with it and to provide information in a timely manner."

Law enforcement agencies have realized the value of obtaining information for use in criminal investigations, as the contents of email, location data, and private SMS messages often can provide the evidence needed to show intent, direct criminal activity, or illustrate that a suspect was in or near a specific location.

Also, the practice of obtaining subpoenas or warrants to obtain personal data from cellphones can be abused, according to George W. Price, a Boston-based attorney with Casner & Edwards LLP who is a former police officer, a former senior special agent with of the U.S. Drug Enforcement Administration (DEA), and a former special assistant district attorney for Middlesex County, MA. "I can find out more about someone through 24 hours of their phone and data use than probably anything else; it's really, really valuable," Price says.

"At the same time, people's expectations of privacy are different than they were 10 or 15 years ago on data devices. You may have a higher expectation now, because you're basically running your whole life through this digital device, which is not necessarily just used for criminal activity. So, I think we're in new territory, as far as how law enforcement can better access that."

"There are very few legal limits on what governments can do with even the most personal data once they have it."

Most technology companies realize the value of protecting personal information, at least within the U.S., where personal privacy is seen as a pillar of the U.S. Constitution. Says Price, "My sense is that [holders of personal data] are not afraid to push back on law enforcement when they feel like there is not enough evidence, or the warrant doesn't meet the proper standards."

Further, there is very little accurate reporting surrounding the effectiveness of warrants used to get private information from users' data in criminal cases, says Stephen Smith, a former federal magistrate judge in Houston, and now Director of Fourth Amendment & Open Courts at Stanford's Center for Internet and Society.

"If somebody would ask me what kind of legislation is most urgently needed right now, I'd say we need a reporting requirement for all these things, similar to what we have for the wiretaps," Smith says, referring to reports provided by federal and state officials on applications for orders for interception of wire, oral, or electronic communications. "We [would] have a complete picture of what's going on and could see how often these techniques are used for child predators, hostage-taking situations, or other really violent crime, versus how many of these are for identity theft, drug possession cases, or run-of-the-mill cases that don't really require the extraordinary means to get this stuff."

Protections on the collection of personal data are far slimmer in jurisdictions outside the U.S. For example, in the U.K., the police can download cellphone data without a warrant, and news reports indicate that cloud extraction technologies provided by

companies such as Petah Tikvah, Israel-based Cellebrite and Alexandria, VA-based Oxygen Forensics can enable law enforcement agencies in the U.K. to continuously track social media accounts, as well as using facial recognition to analyze data extracted from the cloud. U.K. police departments cite three specific powers under which they derive their authority to access this information, including the Police and Criminal Evidence Act 1984 (PACE), the Investigatory Powers Act 2017, or the Regulation of Investigatory Powers Act 2000.

In Japan, the Ministry of Internal Affairs and Communications used to require mobile carriers to obtain the permission of users before sharing any location data with government authorities. However, in June 2015, this requirement was dropped, and news reports indicated some carriers were providing location data to the government, mostly relating to crime investigations. In response, Japanese mobile carrier NTT Docomo announced in May 2016 five smartphone models that would allow authorities to track their locations without users knowing.

Japan has asked owners of both public and private surveillance cameras, as well as wireless carriers, to make user data available to authorities without warrants. This practice, which the Japanese government believes is helpful in solving crimes, as well as tracking domestic abuse cases, is seen as one reason why Japan's crime rate is about a quarter that of the U.S.

The coronavirus outbreak earlier this year has led to even greater sharing of location data between mobile carriers and data collectors. Mobile carriers in Italy, Germany, and Austria shared location tracking info with authorities, while Taiwan, Singapore, and Hong Kong used location monitoring systems to ensure that people who were carrying COVID-19 were staying at home. Further, the Israeli government in March approved emergency measures that allowed its security agencies to track the mobile-phone data of people suspected to be infected with the coronavirus, as well as allowing authorities to enforce quarantines and warn those who may have come into contact with people infected with the virus.

“Realistically, cellphone tracking is already a pretty widespread practice,” says Jennifer Fernick, a technology Fellow at the National Security Institute at George Mason University in Virginia, and the head of research and engineering with NCC Group, a global cyber security and risk mitigation firm based in Manchester, U.K. Fernick notes that in order for a cellphone to work, it must be able to connect to various cell towers, and as the phone connects with a tower, location information can be gathered. “So, to some extent that [location] data is already out there, as it is core to how cellphone networks are designed. To defend against that, there’s not much you can do, other than put your phone in the fridge, or maybe throw it in the ocean.”

Beyond simply using carrier data to track cellphone users, multinational technology companies, including Google, Unacast, Tectonix, X-Mode, and Facebook, among others, are now making available user location data that is captured via apps on users’ smartphones, to track social-distancing efforts. While the data is anonymized—and users that have turned on location tracking have, by default, consented to this information being captured by accepting the terms and conditions of the apps they use—there is a fear that this information may be stored forever by authorities, and used in unrelated matters.

Fred Cate, vice president for research at Indiana University and founding director of the university’s Center for Applied Cybersecurity Research, points to the Health Insurance Portability and Accountability Act of 1996 (HIPAA) and to the laws most U.S. states have “that give states enormous authority when addressing public health issues.

“I don’t doubt for a moment that states have the authority to use this data, which in every case I am aware of they are getting from a third party, in any event. To my mind, the bigger challenge isn’t whether they can get it, but what can they do with it once they have it?”

Cate says that while the Fourth Amendment clearly lays out the proper procedures for obtaining data (generally requiring the government to show it has a legitimate reason for needing

the data), once the government has it, there are no laws covering how long the government may keep the data, or how it may be used in unrelated situations.

“There are very few legal limits on what governments can do with even the most personal data once they get it,” Cate says, noting that in cases involving public safety, security, or health issues, “I suspect almost everyone would approve of the use. But what if, once the government has the data, they then use it for unrelated purposes?”

Cate notes financial information collected during a criminal investigation on money laundering, for example, could be turned over to the Internal Revenue Service if instances of non-related tax evasion activity were found, even in the absence of a criminal charge or conviction. Indeed, given the dearth of regulation, Cate says, there are no usage or time limits to what the U.S. government can do with that data.

“Some agencies have policies,” Cates says. The U.S. Federal Bureau of Investigation (FBI), for example, “used to delete information about you when you turned 70 or 75, but that changed after 9/11. However, that was just an internal policy, and there was no legal force to that. In other words, there’s no time limit on that data.” **■**

Further Reading

The Fourth Amendment to the U.S.

Constitution:
<https://constitutioncenter.org/interactive-constitution/amendment/amendment-iv>

Google’s Process for Handling Requests for User Information: <https://bit.ly/3ahEKQh>

How the U.S. Government is Tracking People via their Cell Phones, *The Wall Street Journal*, Feb. 10, 2020, <https://www.youtube.com/watch?v=SXAShotdFZo>

U.K. Police and Criminal Evidence Act 1984
<http://www.legislation.gov.uk/ukpga/1984/60/contents>

U.K. Investigatory Powers Act 2017
<http://www.legislation.gov.uk/ukpga/2016/25/contents/enacted>

U.K. Regulation of Investigatory Powers Act 2000
<http://www.legislation.gov.uk/ukpga/2000/23/contents>

Keith Kirkpatrick is principal of 4K Research & Consulting, LLC, based in New York, USA.

© 2020 ACM 0001-0782/20/10 \$15.00

ACM Member News

CYBERSECURITY AT OAK RIDGE NATIONAL LABORATORY



“My father brought home a computer when I was in high school, and I taught myself to program on one

of the early TRS-80s,” says Deborah Frincke, associate laboratory director for the National Security Sciences Directorate at the U.S. Department of Energy’s Oak Ridge National Laboratory (ORNL).

Frincke’s early computing experience helped her to discover her passion. She went on to earn her undergraduate, master’s, and doctoral degrees in computer science, all from the University of California, Davis.

After obtaining her Ph.D., Frincke joined academia as a professor of computer science at the University of Idaho. After that, she moved to the Pacific Northwest National Laboratory, where she rose to chief scientist for cybersecurity before leaving for the National Security Agency (NSA).

At the NSA, Frincke served in various roles, most recently as research director. She was also the agency’s Science Advisor, which meant she needed to understand and advise on a diverse range of fields including mathematics, computer science, cybersecurity, quantum and high-performance computing, engineering, and various physical sciences.

Throughout her career, Frincke has focused on cybersecurity, especially collaborative approaches to defensive aspects of cybersecurity, to better protect systems and identify vulnerabilities.

At ORNL, Frincke is assembling a national security science directorate, which requires inventorying current cybersecurity initiatives at the facility, then determining how to strengthen them, and set the overall strategic direction.

“One of the things that excites me about this job is increasing my scope beyond those fields I led at NSA,” Frincke says.

—John Delaney

Fran Allen: 1932–2020

FRANCES E. ALLEN, an American computer scientist, ACM Fellow, and the first female recipient of the ACM A.M. Turing Award (2006), passed away on Aug. 4, 2020—her 88th birthday—from complications of Alzheimer’s disease.

Allen was raised on a dairy farm in Peru, NY, without running water or electricity. She received a BS degree in mathematics from the New York State College for Teachers (now the State University of New York at Albany). Inspired by a beloved math teacher, and by the example of her mother, who had also been a grade-school teacher, Allen started teaching high school math. She needed a master’s degree to be certified, so she enrolled in a mathematics master’s program at the University of Michigan. There she took one of the first courses ever offered in computer programming. Fran interviewed with IBM on campus and took their offered job with the intent of paying off her student loans before pursuing her intended career as a teacher.

IBM announced FORTRAN, one of the first high-level languages, exactly two months before Allen’s arrival in July 1957. She was immediately put to work teaching the language to IBM scientists. To teach herself how the FORTRAN compiler worked, she read its source code. Thus began her interest in compilers. Her “throwaway job,” as she called it at first, proved so compelling

“Fran was a lovely warm person and a strong feminist.”

BARBARA SIMONS
ACM PRESIDENT, 1998–2000

that she stayed 45 years, becoming the first female IBM Fellow in 1989.

Allen worked on compilers for IBM’s first transistorized computer, the IBM 7030 (also known as the Stretch), and the IBM 7950 Harvest, a one-of-a-kind system designed for breaking codes at the U.S. National Security Agency. These were the fastest systems in the world from their introduction 1961 until 1964; NSA used the Harvest until 1976 when its mechanical parts wore out.

IBM’s FORTRAN translated human-readable mathematical formulas and algorithms into machine code, but the resulting programs were significantly larger and slower than what programmers fluent in machine code could produce. Allen created an optimizing compiler with John Cocke (1987 ACM A.M. Turing Award recipient) and the rest of the IBM team so that the compiled code would be worthy of the hardware on which it was running. In addition to FORTRAN, the finished compiler could also handle Autocoder, a business language, and Alpha, a language created for code breaking.

Still determined to be a teacher, Allen became the driving force behind seminal papers in optimizing compilers. Her first, “Program Optimization,” was distributed internally at IBM in 1966 and published in the 1969 *Annual Review in Automatic Programming*. Her 1970 paper, “Control Flow Analysis,” appeared in *SIGPLAN Notices* (July 1970). In 1971, she and Cocke published “A Catalog of Optimizing Transformations,” an IBM technical report that describes “loop transformations,” “redundant sub-expression elimination,” “constant folding,” “dead code elimination,” “instruction scheduling,” and many other techniques that are still used in optimizing compilers.

Allen was also “an enormously kind and encouraging manager,” recalls Paula Newman, a retired computer

scientist who reported to Allen in the late 1960s. “When I saw a letter ... suggesting a somewhat different method of program structure analysis, she allowed me to pursue that approach, and even sent me to an IBM-wide employee appreciation event in Montreal as reward for my work. I believe she continued that management style for the rest of her career.”

These sentiments are echoed by IBM’s current CEO, Arvind Krishna: “Fran spent her life working to advance the field of computing. ... Apart from her technical genius, we remember Fran for her love of teaching and her passion to inspire and mentor others.” To honor Allen and her efforts, IBM established the Frances E. Allen Women in Technology Mentoring Award in 2000.

Allen took sabbaticals at New York University in 1970 and at Stanford University in 1977. “Fran was the only woman professor I had in graduate school,” observed Anita Borg (1949–2003), who founded the Institute for Women and Technology (renamed AnitaB.org in 2017).

Among many honors, Allen was awarded the Computer Society Babage Award in 1997, the Augusta Ada Lovelace Award from the Association for Women in Computing in 2002, and the IEEE Computer Pioneer Award in 2004. Two years later, she was the first woman recipient of the ACM A.M. Turing Award (19 years after her colleague Cocke), “For pioneering contributions to the theory and practice of optimizing compiler techniques that laid the foundation for modern optimizing compilers and automatic parallel executions.”

A member of the National Academy of Engineering and the American Philosophical Society, Allen was also a Fellow of the American Academy of Arts and Sciences, the ACM, the IEEE, and the Computer History Museum. IEEE established the Frances E. Allen medal,

“Apart from her technical genius, we remember Fran for her love of teaching and her passion to inspire and mentor others.”

ARVIND KRISHNA
IBM CEO

to be awarded for the first time in 2022, to honor her career achievements.

“Fran was not swayed by short-term recognition, but instead focused on the significance of the research and technical problems that she worked on and encouraged her team to work on,” notes ACM fellow Vivek Sarkar. She installed a ‘shoot for the moon’ attitude in all of us.”

Allen also loved exploring, climbing mountains in Austria and China, including a 14,000-foot peak in the Himalayas, and traveling across the Arctic without maps or radio contact. She even established a new route across Ellesmere Island in the Arctic Archipelago, the most northerly point of land in Canada.

“Fran was a lovely warm person and a strong feminist,” recalled ACM Fellow and past president Barbara Simons. “I had the pleasure of knowing Fran as a colleague and a friend.” **□**

Simson Garfinkel is the U.S. Census Bureau’s Senior Computer Scientist for Confidentiality and a part-time faculty member at George Washington University in Washington, D.C., USA. He is an ACM Fellow.

Eugene H. Spafford is a professor of computer science and the founder and executive director emeritus of the Center for Education and Research in Information Assurance and Security at Purdue University, W. Lafayette, IN, USA. He is an ACM Fellow.

Copyright held by authors/owners.





DOI:10.1145/3417074

Michael A. Cusumano

Technology Strategy and Management

Self-Driving Vehicle Technology: Progress and Promises

Seeking the answer to the elusive question, 'Are we there yet'?

AUTOMAKERS HAVE ALREADY spent at least \$16 billion developing self-driving technology, with the promise of someday creating fully autonomous vehicles.² What has been the result? Although it seems that we have more promises than actual progress, some encouraging experiments are now under way, and there have been intermediate benefits in the form of driver-assist safety features.

Engineers started on this quest to automate driving several decades ago, when passenger vehicles first began deploying cameras, radar, and limited software controls. In the 1990s, automakers introduced radar-based adaptive cruise control and dynamic traction control for braking. In the 2000s, they introduced lane-departure warning and driver-assist parking technology. Since 2017, Waymo, Uber, Daimler, the U.S. Postal Service, and several other automakers all have launched experiments with robo-taxis or robo-trucks, targeting Level 4 Autonomy (see the sidebar on the last page of this

column).^{4,13} If and when this technology will make its way into your average passenger vehicle is uncertain, but there is no doubt that companies have been moving closer toward their goal.

The basic technologies and engineering skills needed to make self-driving vehicles more widely available already exist. The most popular camera packages from Mobileye (purchased by Intel in 2017) and OmniVision are relatively inexpensive. However, some self-driving systems deploy much more expensive lasers (usually referred to as “lidar” for Light Detection and Ranging) as well as radar and ultrasound sensors, provided by firms such as Ibeo, Velodyne, and Autoliv. Major auto parts and technology suppliers, led by Bosch, Denso, Aptiv (formerly Delphi Automotive, which also purchased the AI and robotics software company NuTonomy in 2017), TRW, and Continental, assemble components into various driver-assist systems and use microprocessors from Intel-Mobileye, Nvidia, and ARM. Blackberry, formerly a pioneer in secure

email and smartphones, has become a player in the automotive IoT software market with its QNX operating system, which runs on some 150 million vehicles.¹¹ Green Hills Software competes in this business as well, along with Google. Automakers, auto parts vendors, and robotics and AI startups, all have been learning how to design the AI and machine-learning applications needed to process data and warn drivers or provide instructions to the vehicle subsystems.

Electric vehicles rely heavily on computers to control their functions, and this characteristic makes them especially suitable for self-driving technology. Not surprisingly, Tesla vehicles deploying driver-assist technology have logged nearly two billion miles and the company probably leads the industry in data collection.¹⁴ Tesla was able to commercialize its Autopilot system because it used cameras, radar, and ultrasound, rather than the more expensive lidar. However, Tesla vehicles are still somewhere between Levels 2 and 3—far from the goal of autonomous driving. They also have been in-



IMAGERY BY PETOVARGA

involved in several high-profile accidents when drivers stopped paying attention, so the company now insists that drivers keep their hands on the steering wheel and eyes on the road.

Another key player is Waymo, founded as a Google R&D project in 2009 and spun off as a fully owned subsidiary in 2016. Waymo does not manufacture cars but has partnered with Fiat-Chrysler, Audi, Toyota, and Jaguar to retrofit their vehicles. It also makes a lot of its own hardware and software to reduce costs. Waymo's technology is more advanced than Tesla, and is presumed to operate at Level 4—but with caveats. The vehicles drive mainly on predefined routes and rely on an expensive combination of lidar, cameras, and radar, as well as human drivers for backup. Since March 2019, Waymo has been operating 600 autonomous vehicles and claims the lead in Level 4 data, with approximately 20 million miles logged on public roads.¹⁶ Since mid-2019, Waymo also has been operating a robo-taxi pilot in California, offering thousands of rides each month.⁷ Al-

though it has lost billions of dollars, Waymo's ultimate goals are to offer ride-sharing services with tens of thousands of vehicles (perhaps with Uber and Lyft as partners) and to license technology to automakers and service providers. To expand its ride-sharing business, Waymo has ordered 62,000 Chrysler Pacifica vans and another 20,000 Jaguar I-Pace cars.¹⁰

Even automakers with modest financial resources can now buy access to self-driving technology. Several companies provide turnkey driver-assist or semi-autonomous driving systems; others focus on data and simulation software, sensor hardware (cameras, lidar, radar, and ultrasound), mapping and location-based software, and vehicle communications systems.⁵ But there is a problem: Exactly what combination of hardware and software works best remains unclear, and there is, as yet, no single industrywide "platform" or common approach for self-driving vehicle technology and communications. Tesla could have been an industry leader by making its software platform available,

but it has proceeded largely on its own.

Other automakers and ride-sharing businesses have formed partnerships that now compete with each other, though they often rely on the same suppliers.⁵ For example, some 80% of vehicles with Advanced Driver Assistance Systems (ADAS) already use Intel-Mobileye cameras, chips, and software.⁶ Volkswagen is at the center of one group built around Argo AI technology, with Ford as a major investor. This alliance has loose or indirect ties to Mercedes-Benz (Daimler), BMW, Toyota, and GM (which bought Cruise Automation in 2016). Other allies are Lyft and Didi. In addition to Argo AI, technology providers include Bosch, Nvidia, Microsoft, Apple, Huawei, Qualcomm, Baidu/Apollo, TomTom, Waymo, and Here (mapping technology). BMW and Mercedes-Benz have a separate alliance, with loose ties to Renault-Nissan, Geely in China, and Audi (a Volkswagen subsidiary). They rely on many of the same suppliers as well as IBM. Toyota leads another group, with ties to GM, Geely, BMW, Mercedes-Benz, and Uber.

Levels of Autonomy in Self-Driving Vehicles

Level 1—Driver Assistance:

The system and the human driver share control; for example, radar-based adaptive cruise control operates the engine and braking, and assists in lane control while the driver steers.

Level 2—Partial Automation:

The system controls vehicle operations such as acceleration, braking, and steering, but drivers must constantly monitor operations and usually need to hold the steering wheel for the autonomous system to operate. Assists steering, lane changing, traffic-jam driving (low-speed version of cruise control), and overtaking.

Level 3—Conditional Automation:

The system controls most vehicle operations, but driver monitoring and intervention are still essential. Drivers may be “hands off” on highways for short periods. Assists lane changes, parking, and traffic-jam driving.

Level 4—High Automation:

The system supports self-driving with no or minimal driver intervention, but primarily in mapped locations. Automated lane changing and other features available.

Level 5—Full Automation:

The system requires no human intervention, as in a robo-taxi or robo-truck.

This table is based on various sources, including: Shuttleworth, J. SAE Standards news: J3016 automated-driving graphic update. *SAE.org News*, January 7, 2019; and Madhavan, R. How self-driving cars work: a simple overview. *Emerj.com*, June 3, 2019.

Mercedes-Benz, which has been working with BMW, Audi, and Bosch, launched another partnership in June 2020 with Nvidia to develop a unique software-defined self-driving architecture by 2024.¹⁵ It is not clear how this effort will impact other Daimler/Mercedes-Benz partnerships. Various automakers and technology vendors, including Intel-Mobileye, are also testing self-driving technology in ride-sharing or ride-hailing ventures while partnering with Uber, Lyft, and Didi—and providing competition to Waymo.

Despite the intensifying competition, there are good arguments for more cooperation. First, the technology remains expensive to develop, especially as companies try to move beyond Level 2. Cameras are necessary to view road signs and traffic lights, but they perform poorly in bad weather. Radar is cheap and able to detect the range and speed of distant objects, but radar images are not as precise as the three-dimensional pictures lidar generates, albeit at considerable expense. Ultrasound or land-based sonar, used extensively in Tesla vehicles, provides a 360-degree view that compensates for camera blind spots and aids in parking, but it can only detect nearby objects and does not replace camera vision.¹²

Second, autonomous driving requires massive amounts of data to refine the vehicle control systems. The more fragmented the market and variations in sensor combinations and algorithms, the less usable data available to any one manufacturer or platform provider.

Third, it could be helpful for vehicles to communicate with each other and with some traffic control systems, as airplanes do. This type of communication does not solve the problem of a pedestrian suddenly appearing in front of a moving car, but it could reduce accidents with other vehicles, especially if we retrofit older cars and trucks with inexpensive communications devices or smartphone cameras.¹ The Networking for Autonomous Vehicles Alliance (see <https://navalliance.org/>), founded by Bosch, Continental, Marvell, NVIDIA, and Volkswagen of America, is also working “to provide a platform for the automotive industry to develop the next generation of in-vehicle network infrastructure for autonomous vehicles,” though it has yet to set any global standards.⁹

Ride-sharing and ride-hailing companies are likely to buy or lease fleets of self-driving vehicles to get rid of driver costs, their main expense, and their large-scale purchases could help reduce costs for the automakers. However, the ride-sharing companies are already losing billions of dollars per year, and they will have to take on the enormous costs of owning or leasing millions of their own vehicles.^a Self-driving technology might even someday eliminate demand from companies like Uber, Lyft, and Didi.³ Most privately owned automobiles sit idle approximately 95% of the time.⁸ Tesla and other auto-

makers are exploring how to enable owners to share their vehicles when not in use and earn some revenue from this activity, rather than relying on Uber or other intermediaries.

Conclusion

In sum, there currently are several experiments with robo-taxis and robo-trucks on prescribed routes, but still with human drivers as backups. Full automation at Level 4 or 5 remains a distant goal for the average consumer, and it is difficult to pinpoint a timeframe when this will become a reality. Meanwhile, all this R&D is not for naught. Even if automakers never advance much beyond Level 3 over the next decade, driver-assist technology has already made driving safer. Assisting rather than replacing drivers should perhaps be our end goal, rather than full automation. **C**

a See M.A. Cusumano, ‘Platformizing’ a bad business does not make it a good business, *Communications* (Jan. 2020).

References

1. Abuelsamid, S. iOnRoad collision alerts? There's an app for that. *Motor Trend* (Feb. 16, 2012).
2. Baldwin, R. Self-driving-car research has cost \$16 billion. What do we have to show for it? *Car and Driver* (Feb. 20, 2020).
3. Cusumano, M.A., Gawer, A., and Yoffie, D.B. *The Business of Platforms: Strategy in the Age of Digital Competition, Innovation, and Power*. (2019), 223–226.
4. Davies, Alex. Self-driving trucks are now delivering refrigerators. *Wired* (Nov. 13, 2017).
5. Firstmile VC. Decoding the autonomous driving landscape. *Medium.com*. (July 31, 2019).
6. Gedalyahu, D.B. Intel: Mobileye is our fastest growing business. *En.Globes.Co.II* (Nov. 6, 2019).
7. Korosec, K. Waymo's robotaxi pilot surpassed 6,200 riders in its first month in California. *TechCrunch* (Sept. 16, 2019).
8. Morris, D. Z. Today's cars are parked 95% of the time. *Fortune*. (Mar. 16, 2016).
9. NAV Alliance. NAV alliance picks up speed with new members. Press release, September 23, 2019.
10. Rushe, D. 'I'm so done with driving': is the robot car revolution finally near? *The Guardian*, March 10, 2019.
11. St. John, A. BlackBerry CEO talks competitors, autonomy at CES. *Automotive News*. (Jan. 8, 2020).
12. Tsyktopr, V. LIDAR vs radar vs sonar: Which is better for self-driving cars? *Cyberpulse* (May 28, 2018).
13. TU-Automotive. Robo-trucks are where the self-driving revolution begins. *IoTWorldToday.com* (May 28, 2019).
14. Udayan, T. Autonomous vehicle technology companies to watch out for. *IoTforAll* (Feb. 4, 2020).
15. Voigt, A. Mercedes-Benz and Nvidia partner on autonomous driving—numerous thoughts and questions. *CleanTechnica* (June 23, 2020).
16. Wiggers, K. Waymo's autonomous cars have driving 20 million miles. *VentureBeat* (Jan. 6, 2020).

Michael A. Cusumano (cusumano@mit.edu) is Deputy Dean and SMR Distinguished Professor at the MIT Sloan School of Management, and co-author of *The Business of Platforms* (2019).

The author thanks Annabelle Gawer and David Yoffie, as well as the *Communications* Viewpoints co-chairs, for their comments.

Copyright held by author.

Inside Risks

A Holistic View of Future Risks

Almost everything is somehow interrelated with everything else—and that should not surprise us.

THIS COLUMN CONSIDERS SOME challenges for the future, reflecting on what we might have learned by now—and what we systemically might need to do differently. Previous Inside Risks columns have suggested that some fundamental changes are urgently needed relating to computer system trustworthiness.^a Similar conclusions would also seem to apply to natural and human issues (for example, biological pandemics, climate change, decaying infrastructures, social inequality), and—more generally—being respectful of science and evident realities. To a first approximation here, I suggest almost everything is potentially interconnected with almost everything else. Thus, we need moral, ethical, and science-based approaches that respect the interrelations.

Some commonalities across different disciplines, consequent risks, and what might need improvement are considered here. In particular, the novel coronavirus (COVID-19) has given us an opportunity to reconsider many issues relating to human health, economic well-being (of individuals, academia, and businesses), domestic and international travel, all group activities (cultural, athletic, and so forth), and long-term survival of our planet in the



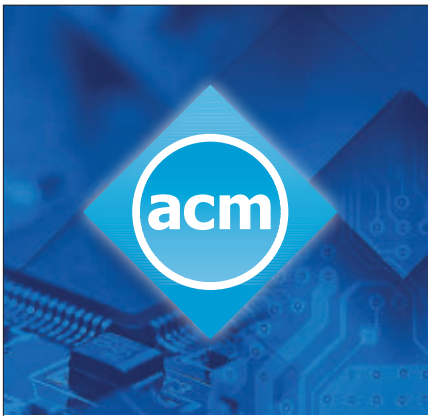
face of natural and technological crises. However, there are also some useful lessons that might be learned from computer viruses, malware, and inadequate system integrity, some of which are relevant to the other problems—such as computer modeling and retrospective analysis of disasters, supply-chain integrity, and protecting whistle-blowers.

A quote from Jane Goodall in an interview in April 2016 seems more broadly relevant here than in its original context: “If we carry on with business as usual, we’re going to destroy

ourselves.” The same is true of my quote from the early crypto wars regarding export controls: “Pandora’s Cat Is Out of the Barn, and the Genie Won’t Go Back in the Closet.” We are apparently reaching a crossroads at which we must reconsider potentially everything, and especially how it affects the future.

Priorities Among Competing Goals
Human civilization does not tend to agree among issues such as *fairness, equality, safety, security, privacy, and self-determination* (for example). With

^a For example, see “How Might We Increase System Trustworthiness?” *Communications* (Oct. 2019); <http://www.csl.sri.com/neumann/cacm247.pdf>



Advertise with ACM!

Reach the innovators and thought leaders working at the cutting edge of computing and information technology through ACM's magazines, websites and newsletters.



Request a media kit with specifications and pricing:

Ilia Rodriguez
+1 212-626-0686
acmm mediasales@acm.org



COVID-19, economical well-being, health care, climate change, and other issues (some of which are considered here), if we cannot agree on the basic goals, we will never reach whatever they might have been—especially if the goals appear to compete with each other.

The Importance of Fundamental Principles

Numerous principles for computer system security and integrity have been known for many years, and occasionally practiced seriously. Some corresponding principles might be considered more broadly in the combined context of risks in engineering computer-related systems, but also in natural systems.

Albert Einstein wrote “It can scarcely be denied that the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience.”^b This is often paraphrased as “Everything should be made as simple as possible, *but no simpler.*” Although the longer statement could be thought of as applicable to trying to explain things as they are (for example, the universe), the simplified version (“should be made”) is also fundamental to the development of new computer systems, as well as in planning proactively for potential catastrophes and collapsing infrastructures.

This *principle*, together with principles relating to transparency, accountability, and scientific integrity, suggest dealing openly and appropriately with risks, while being respectful of science and reality throughout. For example, we tend to make huge mistakes by stressing short-term gains (particularly financial), while ignoring the long-term risks (including everything else). Unfortunately, the gains are unevenly distributed, as the rich get richer, and the poor tend to get poorer and suffer much more.

The principles relating to completeness are particularly critical to computer system design, implementation, applications, and human interfaces, but also regarding responses to

pandemics, climate change, and the planet’s environment—along with their implications for human health and well-being, and resource exhaustion of rare elements.

A closely related principle of *pervasive holism* invokes a big-picture view of the Einstein principle, in which everything is potentially in scope unless explicitly ruled out—for example, for reasons of impossibility, feasibility, or perhaps for mistaken decisions about costs, when the long-term overall benefits would dramatically outweigh the short-term savings. Pervasive holism represents the ability to consider all relevant factors, and the ensuing risks. It is relevant broadly across many disciplines. For example, it is essential in the design of computer-communication systems. It encourages systems to be designed to compensate for a wide range of threats and adversities, including some that might not be anticipated *a priori*. Similarly, climate change is causatively linked with extreme weather conditions, melting glaciers, more disastrous fires, human activities, fossil fuels, changes in agriculture, and—with nasty feedback—greater demands for air conditioning and refrigerants such as hydrofluorocarbons that are making the problems worse. On the positive side, atmospheric and sea changes have been observed during the pandemic shutdown (with reduced fuel consumption and much less travel), reinforcing arguments that alternatives to fossil fuels are urgently needed (especially as they are becoming increasingly economical and competitive).

Numerous principles for computer system security and integrity have been known for many years, and occasionally practiced seriously.

^b See <https://quoteinvestigator.com/2011/05/13/einstein-simple/>

Many nations have clearly realized that careful application of scientific analysis is always desirable, but it can be misused or misapplied. In confronting pandemics, massive immunization programs must be preceded by extensive testing, without which they can have serious consequences (including organ failures, deaths, iatrogenic effects, and in some cases allergic reactions such as anaphylaxis). In pharmaceuticals, some effects are disingenuously called ‘side-effects’—whereas in many cases these effects are well known to have occurred (and are often extensively enumerated in the labeling). However, the effects of deforestation, pesticides, toxic environments (water, air, polluted oceans), non-recyclable garbage, overuse of antibiotics, and so on should by now all be well recognized as long-term risks.

In today’s novel coronavirus and its ongoing mutations, a holistic approach requires anticipating human physical and mental health factors, and their interactions with economic factors and social equality (all persons are supposedly created equal, but usually not treated accordingly—but what about other creatures?), along with future implications, globally rather than just locally. It also requires understanding potential long-term damage—for example, effects on heart, brain, and other organs are still unknown. Fully anticipating the consequences of insurance policies that would not allow existing preconditions is also a major issue, in light of the huge numbers of COVID-19 infections worldwide. Equality in almost everything is desirable, especially in education when home schooling is impossible, broadband access is spotty or nonexistent, and the lack of ubiquitous Internet-accessible devices is a show-stopper for many children. Equal opportunity to vote is also critical, but is being badly abused. Furthermore, spreading disinformation and other forms of disruption can be especially damaging in all of the preceding cases. So, many of these issues are actually interrelated. As one further example of the extent of interrelationships and interlocking dependencies, the realization that arctic glacial melting is releasing methane and possibly ancient viruses from earlier pandemics is also relevant.

Adherence to ethical principles is of course also likely to contribute to human integrity, as well as to transparency, accountability, and reality.

Principles involving controllability, adaptability, and predictability require better understanding of the importance of *a priori* requirements, as well as the vagaries of models, designs, development, implementation, and situational awareness in real time. These are vital in computer system development. In pandemics, these principles should help reduce the uncertainties of taking different approaches to limiting propagation of contagion, severity of cases, duration of disruption, extent of acquired immunities, and above all a willingness to accept reality and scientific knowledge.

A caveat is needed here: The preceding principles can be used effectively by people who deeply understand the fields in which they are working—and who also have a willingness to work well with colleagues with a better understanding of other areas. In the absence of such knowledge and willingness, the principles are likely to be very poorly misapplied. Humility is a virtue in this regard.

Legal and Ethical Principles

In the social and economic arena, there is a similar need for close attention to the core legal principles driving privacy, antitrust, labor rights, environmental damage, and so on. There is a small but growing group of legal scholars who are revisiting our legal foundations, in an overarching framework they call an approach to ‘law and political economy’, somewhat in reaction to the very influential ‘law and economics’

approach that originated from the University of Chicago.^c

Adherence to ethical principles is of course also likely to contribute to human integrity, as well as to transparency, accountability, and reality.

Models, Predictions, and Planning

Creating realistic models for computational or other problems considered here is always an art form. A selected model may itself be fundamentally divergent from reality. Assumptions made may be speculative, or in some cases intentionally biased to enable the model to justify preconceived goals. (With statistics, anything can be ‘proven’.) Furthermore, static models are unable to adapt to changing events, so the model must be adaptable to evolving realities and the emergence of better knowledge. This is true of pandemics, climate change, as well as computer system behavioral modeling.

Having well-designed models that provide transparency, respect reality, and are mathematically sound is very important—in order to be able to reason sensibly. However, because models inherently represent abstractions of reality, reasoning about models typically introduces discrepancies between the models and reality. Predicting the future based on erroneous models and erroneous logic is not a path to success. Similar remarks apply to statistical analysis of inherently multidimensional problems. These issues have clearly been raised in predicting the progress of pandemics, climate change, and the trustworthiness of computer systems (for example). Although this is a particularly fundamental area, it deserves much more study. However, when evidence clearly demonstrates poor results, it is time to reassess failed remediations.

Testing and Verifying

Testing finds problems, but cannot find the absence of problems. Verification can find some of the problems, but many others are beyond routine analysis—such as side channels, hardware attacks, and other things that might not be included in threat models. Thus,

^c See https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3547312



Association for
Computing Machinery

2018 JOURNAL IMPACT
FACTOR: 6.131

ACM Computing Surveys (CSUR)

ACM Computing Surveys (CSUR) publishes comprehensive, readable tutorials and survey papers that give guided tours through the literature and explain topics to those who seek to learn the basics of areas outside their specialties. These carefully planned and presented introductions are also an excellent way for professionals to develop perspectives on, and identify trends in, complex technologies.



For further information
and to submit your
manuscript,
visit csur.acm.org

even a combination of both may not be enough, which applies to computer algorithms, protocols, software, and hardware, but also to some of the other areas considered here. For example, biological testing and applications of artificial intelligence and deep learning need to have a sounder basis that could eliminate vastly too many false positives and false negatives, as well as other forms of unrealistic results.

Formal methods are increasingly being applied to software hypervisors (for example, CertiKOS, seL4, and the Green Hills Integrity Multivisor) and to hardware (for example, CHERI, Centaur). Formal modeling of biological pathways, techniques to stimulate immunities, effects of climate change, short- versus long-term consequences, and many other possible approaches could be considered using formal analysis. Of particular interest might be formal analysis of requirements, models, and analysis techniques in other areas considered here.

System and Supply-Chain Integrity

The availability and integrity of delivered computer-related systems and medical supplies clearly present enormous problems, which are likely to be exacerbated in times of crisis. Over recent years, the implementation of many entities has increasingly been outsourced and off-shored, including computer hardware fabrication, just-in-time delivery of automobile parts, hospital health-care necessities, and even food. It should be obvious that our computer systems, medical supplies, and other resources may be inadequately protected from supply-chain disruptions, tampering, and even fraud.

The effects of monopolized industrial sectors are notable here. The concentration of industrial production of various types in a few very large players (most prominently and visibly in the tech sector, but it is pervasive across the board with hidden monopolies galore, for example, in pharmaceuticals and health-care management).^d It effectively creates fewer but far more consequential

points of failure. In addition, due to the lack of alternatives, product quality suffers when market power is abused, due to the lack of alternatives.^e

Principles of robust and resilient system design, both industrial and computer related, suggest that having many distributed and roughly commensurate producers or protocol participants is preferable to highly centralized structures. The latter might be superficially more efficient, but could mask dramatic failure modes. This notion also shows up in ecology, where diverse and vibrant farm ecosystems are typically more resilient than crop monocultures. Furthermore, concentrated economical power embodied by monopolies is easily converted into political power, leading to contribution-favoring legislation and rising economic inequality.

The advantages of having diverse and widely dispersed (but well coordinated and carefully monitored) actors seem to be preferable in improving distributed computer-system resiliency, economies of industrial organizations, approaches to pandemics, and thriving ecosystems.

System Integrity

Overall system integrity is also an issue. For example, election integrity is dependent not just on voting machines and paper ballots. It also depends on the trustworthiness of registration databases, tabulation and auditing processes, as well as (for example) the avoidance or tolerance of distorting effects such as gerrymandering, selective disenfranchisement, and rampant use of disinformation. Some efforts require effective national leadership, and in some cases extensive international cooperation.

Protecting Protectors and Truth of Information

Reporting of systemic flaws newly found by white-hat hackers has generally become carefully managed in order to avoid flagrant misuse; however, the market for zero-day flaws remains lucrative. Whistle-blowers

^d See <https://prospect.org/health/hidden-monopolies-raise-drug-prices/>

^e See <https://prospect.org/power/monopoly-misrepresentation-and-malpractice-3m-earplugs/>

Willingness to accept and respond to reality is fundamental to avoiding risks.

have often subsequently been victims of character assassinations, particularly those that are fabricated to distract from would-be exposure of misdeeds. Also, numerous medical experts who have dealt with legitimate scientific evidence regarding COVID-19 have been treated as illegitimate purveyors of fake information, as if they had been whistle-blowers spreading false accusations. The same is true of climate change, which requires careful consideration of the underlying science. Conspiracy theories continue to appear. The principles of transparency and accountability are particularly important in these contexts.

Privacy and Related Concerns

Respecting personal privacy is a ubiquitous challenge in every computer-related activity, particularly in the presence of overreaches in widespread surveillance, the desire for cryptographic backdoors for law enforcement, and detailed statistical reporting. In addition, addressing rampant disinformation and hate speech, as well as attacks on whistle-blowers, are in conflict while trying to protect free speech. Some of these and other issues are particularly relevant to pandemics (for example, with the need for intensive monitoring and large-scale fine-grained contact tracing)—as well as almost everything involving big data.

What Is Missing from This Conceptual Big-Picture View?

The discussion here may seem somewhat disconnected, and the desire for holistic approaches overly ambitious. However, it is becoming ever clearer that the topics considered here are

interrelated in ways that are sometimes not obvious. For example, man-made disruptions of nature seem to be biting back at us in various ways, including climate change, health, eco-balance, pollution, and animal-human crossovers of pandemics. This column is merely a high-level attempt to find commonality in what may once have appeared to be disparate subjects. Putting all the pieces together with adequate foresight presents major challenges. Everything along the way needs precise definitions, descriptions, specifications, and logical thought, including the dependencies among the constituent elements. Well-defined realistic abstractions are important, along with well-defined refinements that can be used to determine overall consistency and predictable results. Only then can rational conclusions be reached that have any bearing on reality.

The sense of composing the pieces with predictable assurance is conceptually understood in theory with respect to computer systems, although not often observed in practice. A goal here would be to mirror such approaches with respect to other areas, such as biological processes, pandemic spreading, environmental problems, and other socioeconomic issues, to give them a more scientific and logical basis. Understanding the legal foundations of markets and social interactions is also a basic part of what needs to be included in the holistic view, along with the technological, engineering, and other scientific principles. Identifying any common abstractions and their potential interactions could be very helpful.

In the opposite direction, what might computer technology learn from the ongoing natural-world problems noted here? For example, our system models and risk models for trustworthy computer systems generally fail to consider the risks holistically—for example, neglecting those that are external to the technology. Predicting any consequences on the basis of questionable models is also a major risk, especially if the results superficially seem generally believable—and what one might like to believe. Thus, we need to learn more from each other.

Conclusion

Willingness to accept and respond to reality is fundamental to avoiding risks. The unknown unknowns are always risky, but can be minimized somewhat by proactively seeking to identify the potential risks, and reflecting on Murphy's Law—rather than ignoring the emergence of certain presumed rare disasters that have been emerging much more often, which deserve *a priori* attention (rather than relying on case-by-case *a posteriori* remediation).

This clearly applies to infrastructures, supply chains, and medical preparedness, among other topics considered here—or further topics that could have been included here but were not even mentioned, with the unfortunate consequence of making the discussion *too simple*, in conflict with the Einstein principle.

As has been frequently noted, but which is nevertheless highly relevant here, We Are All In This Together, and Almost Everything Is Increasingly Becoming Interrelated—for better or for worse. Isolated defensive actions have very limited value; your own actions can affect others. Retrogressive governmental actions are counterproductive. Biological viruses and computer risks can both propagate globally with amazing rapidity. In any event, you must protect yourself, while also respecting the well-being of others. Wearing a mask and isolating yourself are akin to being intensely security-aware with respect to computer viruses and phishing attacks, having backups to defend against ransomware attacks, and being cognizant of reality.

Ultimately, more altruistic foresight could help to avoid all sorts of undesirable events, such as pandemics, climate change, environmental disasters, global extinction of species, disparities in education and economic well-being, and unnecessary losses of human life—as well as crossover combinations of these (for example, as varied as the Deepwater Horizon fiasco, deforestation of the Amazon, the demise of honey bees, and wars). And yet, this brief summary is only a beginning. ☐

Peter G. Neumann (neumann@osl.sri.com) is Chief Scientist of the SRI International Computer Science Lab, and has moderated the ACM Risks Forum since its beginning in 1985. He is grateful to Prashanth Mundkur and Tom Van Vleck for helping considerably enrich the holistic perspective in this column.

Copyright held by author.



George V. Neville-Neil

DOI:10.1145/3417099

Article development led by [acmqueue](https://queue.acm.org)
queue.acm.org

Kode Vicious Sanity vs. Invisible Markings

Tabs vs. spaces

Dear KV,

My team resurrected some old Python code and brought it up to version 3. The process was made worse by the new restriction of not mixing tabs and spaces in the source code. An automatic clean-up that allowed the code to execute by replacing the tabs with spaces caused a lot of havoc with the comments at the ends of lines. Why does anyone make a language in which white space matters this much?

White Out

Dear White,

Ever edited a `Makefile`? Although there is a long tradition of the significant use of white space in programming languages, all traditions need to change. In Python, many people have taken issue with the choice to have white space—and not braces—to indicate the limits of blocks of code, but since the developers did not change their minds on this with version 3 of Python, I suspect we are all stuck with it for quite a bit longer, and I am quite sure that there will be other languages, big and small, where white space remains significant.

If I could change one thing in the minds of all programming language designers, it would be to impress upon them—forcefully—the idea that anything that is significant to the syntactic or structural meaning of a program must be easily visible to the human

reader, as well as easily understood by the systems used by developers.

Let's deal with that last point first. Making it easy for tools to understand the structure of software is one of the keys to having tools that help programmers prepare proper programs for computers. Since the earliest days of software development, programmers have tried to build tools that show them—before the inevitable edit-compile-test-fail-edit endless loop—where there might be issues in the program text. Code editors have added colorization, syntax highlighting, folding, and a host of other features in a desperate, and some might say fruitless, attempt to improve the productivity of programmers.

When a new language comes along, it is important for these signifiers in the code to be used consistently; otherwise your editor of choice has little or

no ability to deploy these helpful hints to improve productivity. Allowing any two symbols to represent the same concept, for example, is a definite no-no. Imagine if you could have two types of braces to delineate blocks of code, just because two different parts of the programming community wanted them, or if there were multiple syntactic ways to dereference a variable. The basic idea is there must be one clear way to do each thing that a language must do, both for human understanding *and* for the sanity of editor developers. Thus, the use of invisible, or near-invisible, markings in code, especially tabs and spaces, to indicate structure or syntax.

Invisible and near-invisible markings bring us to the human part of the problem—not that code editor authors are not human, but most of us will not write new editors, though all of us will use editors. As we all know,

U+1F4A9

once upon a time computers had small memories and the difference between a tab, which is a single byte, and a corresponding number of spaces (8) could be a significant difference between the size of source code stored on a precious disk, and also transferred, over whatever primitive and slow bus, from storage into memory.

Changing the coding standard from eight spaces to four might improve things, but let's face it, none of this has mattered for several decades. Now, the only reason for the use of these invisible markings is to clearly represent the scope of a piece of code relative to the pieces of code around it.

In point of fact, it would be better to pick a single character that is not a tab and not a space and not normally used in a program—for example, Unicode code point U+1F4A9—and to use that as the universal indentation character. Editors would then be free to indent code in any consistent way based on the user's preferences. The user could have any number of blank characters used per indent character—8, 4, 2, some prime number, whatever they like—and programmers could choose their very own personal views of the scope. On disk, this format would cost only one character (two bytes) per indent, and if you wanted to see the indent characters, a common feature of modern editors, you flip a switch, and voila, there they all are. Everyone would be happy, and we would finally have solved the age-old conundrum of tabs vs. spaces.

KV



Related articles
on queue.acm.org

File-System Litter

Kode Vicious

<https://queue.acm.org/detail.cfm?id=2003323>

A Generation Lost in the Bazaar

Poul-Henning Kamp

<https://queue.acm.org/detail.cfm?id=2349257>

Demo Data as Code

Thomas A. Limoncelli

<https://queue.acm.org/detail.cfm?id=3355565>

George V. Neville-Neil (kv@acm.org) is the proprietor of Neville-Neil Consulting and co-chair of the ACM *Queue* editorial board. He works on networking and operating systems code for fun and profit, teaches courses on various programming-related subjects, and encourages your comments, quips, and code snips pertaining to his *Communications* column.

Copyright held by author.



Association for
Computing Machinery

ACM Transactions on Computing for Healthcare (HEALTH)

*A multi-disciplinary journal for
high-quality original work on how
computing is improving healthcare*

ACM Transactions on Computing for Healthcare (HEALTH) is the premier journal for the publication of high-quality original research papers, survey papers, and challenge papers that have scientific and technological results pertaining to how computing is improving healthcare.



For further information and to submit
your manuscript, visit health.acm.org

Viewpoint

We Need to Automate the Declaration of *Conflicts of Interest*

Leveraging existing data sources to improve the declaration and management of authorship conflicts of interest.

OVER THE LAST 70 years of computer science research, our handling of conflicts of interest has changed very little. Each paper's corresponding author must still *manually* declare all their co-authors' conflicts of interest, even though they probably know little about their most senior co-authors' recent activities. As top-tier conference program committees increase past 500 members, many with common, easily confusable names, PC chairs with thousands of reviews to assign cannot possibly double-check corresponding authors' manual declarations against their paper's assigned reviewers. Nor can reviewers reliably catch unreported conflicts. Audits at recent top-tier venues across several areas of computer science each uncovered more than 100 instances where, at the first venue, a pair of recent co-authors failed to declare their conflict of interest; at the second venue, someone was assigned to review a recent co-author's submission; and at the third venue, someone reviewed a submission written by a prior co-author from any year. Even the concept of a conflict deserves closer scrutiny: an audit at yet another recent top-tier venue edition found more than 100 cases in which prior co-authors from any year reviewed each other's submissions.

These are issues of scale. Seven-



ty years of exponential growth have turned our little village into a metropolis, and our handling of conflicts of interest (*conflicts* for short) has not kept pace with our community's growth. But we computer scientists are experts at scaling up! We have already addressed issues of scale in many other aspects of our review processes, including enhancements such as double-blind review, multiple submission deadlines, opportunities for revision and rebuttal, and online submission and review management systems.

It is time for our venues to leverage existing data sources to improve the

declaration and management of conflicts, as follows.

1. Uniquely identify all authors in bibliographic data sources, as well as all authors, reviewers, and meta-reviewers in manuscript management systems. (Meta-reviewers are those who manage the review process, such as editors-in-chief and program committee chairs.)

Duplicate names already make it impossible to unambiguously identify by name those involved in the review process, and even make it difficult for conference organizers to ensure they are inviting the right people to join

their program committees. Fortunately, authenticated ORCID^a exist for exactly this purpose, and we should require their use.

2. Disallow changes in the author list after submission. Conflict declarations are based on the author list at the time of submission; subsequent changes may introduce new conflicts not considered during reviewer assignment.

3. Require automated reporting of all observable conflicts. PC chairs can use a service that identifies all conflicts observable in publicly available information on co-authorships, institutional affiliations, and advisor relationships, as explained here.

4. Require authors to self-report only non-observable conflicts, such as new employers, new collaborations, family members, and friends.

5. Automatically audit self-reports in retrospect and share the results with the venue's sponsor or publisher, which should have the power to examine all data they consider relevant and to impose appropriate sanctions for serious violations.

6. Use an independent and conflict-of-interest-free committee to select best papers.

7. Consider the use of a more sophisticated definition of conflict of interest, as explained here.

8. Involve the community and our professional societies as needed, as discussed here.

To see how an automated conflict reporting service for manuscript management systems can work, consider the traditional definition of a conflict: two people have a conflict if they wrote a paper together in the past two years, are at the same institution, are close relatives or friends, were advisor and advisee, or worked together closely on a project in the past two years. Bibliographic databases such as Google Scholar^b and DBLP^c implicitly provide a graph of the relevant co-authorship relationships, and can also be mined with high accuracy to identify advisor-advisee relationships.² DBLP already uses data-driven

It is time for our venues to leverage existing data sources to improve the declaration and management of conflicts.

disambiguation^d of individuals and associates authors with ORCID^a and employers; see, for example, how DBLP handles its 218 different Wei Wangs.^e Authenticated employer information (including unique IDs for institutions) and educational affiliations are also available directly from the ORCID service, and perhaps authenticated advisor information eventually as well.

The conflict service's input is: for each paper, the set of (uniquely identified) authors; the set of reviewers and meta-reviewers, also uniquely identified; and a menu-style specification of the venue's conflict policy. For each paper, the conflict service returns the paper's conflicts, that is, all reviewers and meta-reviewers who have an observable conflict with an author of the paper, along with an explanation of the source of the conflict. These conflicts must be added to the self-reports in the submission system, after which conference organizers can use any method of assigning papers to reviewers, for example, manually, based on bids, or using the Toronto Paper Matching Service.¹ As usual, the assignment algorithm will automatically avoid all review assignments that involve a conflict. Note that the conflict service need not learn anything about a venue's submissions, beyond the set of all authors.

Two standalone beta versions of conflict services are already available to PC chairs, driven by DBLP data; one^f requires authors and reviewers to be

uniquely identified, and the other^g requires only reviewers to be uniquely identified.³ In the longer run, we recommend that outcalls to a conflict service be directly supported by manuscript management systems, so that the system can automatically invoke the conflict service to augment self-reports of conflicts before reviewers are assigned. We also recommend that the authors of reviewer assignment algorithms extend them to avoid additional more subtle biases in the review process, by ensuring diversity of institutions, localities, and countries of origin. Computer science research is now a global enterprise, and we should take advantage of that diversity throughout the review process.

Villagers might not need to lock their doors, but metropolis dwellers would be foolish not to. As village life slowly gave way to the anonymity of the big city, our community has had to establish ethics committees and codes of ethics, policies on plagiarism, authorship, sexual harassment, and so on. Automated reporting of observable conflicts will greatly reduce the big-city crimes of impersonating others and deliberately underreporting conflicts. Automated audits will offer a further deterrent once the conflict service is integrated into submission systems: the system can automatically recompute the observable conflicts some months after the submission deadline and compare them to those stored in the system. At a minimum, missing self-reports should result in a warning letter.

Currently, conflicts are all-or-nothing; today two recent co-authors absolutely cannot review each other's papers, but maybe tomorrow they absolutely can. Big-city life demands a more nuanced definition that recognizes all the shades of gray, so let us acknowledge that conflicts differ in their severity, drop the binary definition of conflict, and define a (degree of) conflict as a real number in $[0, 1]$ computed by a formula specified by the publication venue (the aforementioned menu-style specification). Then we can differentiate between the severity of a conflict and a venue's publicized threshold for automatically disqualifying a reviewer, which will legitimately differ between venues (for example, a

a The Open Researcher and Contributor ID (ORCID) is an international non-profit initiative to uniquely identify scientific and other academic authors; see <https://orcid.org>

b See <https://google.scholar.com>

c See <https://dblp.org>

d See <https://dblp.uni-trier.de/faq/17334571.html>

e See <https://dblp.uni-trier.de/pers/hd/w/wang:wei>

f See <https://github.com/ebina1/conflict-of-interest>

g See <https://www.ntu.edu.sg/home/assourav/research/DARE/closet.html>



Association for
Computing Machinery

ACM Transactions on Evolutionary Learning and Optimization (TELO)

ACM Transactions on Evolutionary Learning and Optimization (TELO) publishes high-quality, original papers in all areas of evolutionary computation and related areas such as population-based methods, Bayesian optimization, or swarm intelligence. We welcome papers that make solid contributions to theory, method and applications. Relevant domains include continuous, combinatorial or multi-objective optimization.



For further information
and to submit your
manuscript,
visit telo.acm.org

The prototypes mentioned here show that one can already build useful standalone conflict services that rely on readily available data.

workshop versus a top-tier conference). The conflict service described here can easily support such venue-specific cut-off scores and real-valued functions for computing conflicts, making it easy for venues to define and experiment with more sophisticated measures.

We also need to recognize that multiple co-authorships indicate a stronger tie. A dozen papers co-authored five years ago may pose as much of a conflict as does a single paper co-authored last year, because those dozen papers indicate a very strong tie. Further, conflicts can have multiple contributing facets, for example, same institution, same city, or a highly overlapping set of prior co-authors. We can weight each type of tie between researchers according to the strength of their tie, model the fading of ties over time as a continuous function, and devise a method to gracefully combine multiple weighted and faded factors into an overall conflict score, corresponding to our best estimate of the chance that two people cannot impartially review each other's work.

The prototypes mentioned here show that one can already build useful standalone conflict services that rely on readily available data. But we will need greater community involvement to reach the ultimate solution. Beyond the steps outlined that each venue can take today, we advocate four steps at the community level.

1. To reach a solution suitable for all of computer science, our community will need to provide coordination and funding for infrastructure construction. This could come from the ACM Publications Board, the SIG Governing Board, the IEEE Technical Activities Board, ACM and/or IEEE as a whole, or

even a computing-wide consortium that includes non-profit societies and for-profit publishers.

2. To expand the definition of a conflict and devise the infrastructure to support that definition, we may need input from experts on the social issues of privacy and security; the technical issues of data collection, organization, and maintenance; the policy issues inherent in defining conflict broadly yet specifically; and the administrative issues in long-term maintenance and evolution of a conflict service.

3. We should encourage research into relevant topics, including definitions of conflict, scalable algorithms to identify conflicts, and sources and methods for handling suspected false positives.

4. Once they are in place, we should share our community's metrics, mechanisms, and infrastructure with the global research enterprise, including other scientific disciplines and the National Academies of interested countries.

Life in the big city poses new threats and challenges, but we can leverage the metropolis's great infrastructure to address those problems. By taking advantage of existing datasets, services, and mining algorithms, we can eliminate almost all the tedium of declaring and managing conflicts, with the pleasant side effect of reducing the metropolitan crime rate. With those measures in place, we can move on to develop a more nuanced understanding of what constitutes a conflict of interest. □

References

1. Charlin, L. and Zemel, R.S. The Toronto paper matching system: An automated paper-reviewer assignment system. In *Proceedings of the International Conference on Machine Learning (ICML)* 2013.
2. Wang, C. et al. Mining advisor-advisee relationships from research publication networks. In *Proceedings of the 16th ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.
3. Wu, S. PISTIS: A conflict of interest declaration and detection system for peer review management. In *Proceedings of the 2018 ACM SIGMOD/PODS Conference*, 2018.

Richard T. Snodgrass (rts@email.arizona.edu) is a Professor and Galileo Scholar at the University of Arizona, Tucson, AZ, USA. He is an ACM Fellow, has served as editor-in-chief of ACM TODS and as chair of ACM SIGMOD and the ACM Publications Board, and was founding co-chair of the ACM History Committee.

Marianne Winslett (winslett@illinois.edu) is a research professor emerita at the University of Illinois, Urbana IL, USA. She is an ACM Fellow and has served as a coeditor-in-chief of ACM TWEB, as an officer of SIGMOD and SIGART, on the steering committees of ACM CIKM and ACM CCS, and on the editorial boards of ACM TODS, ACM TISSEC, ACM TWEB, IEEE TKDE, and the VLDB Journal.

Copyright held by authors.

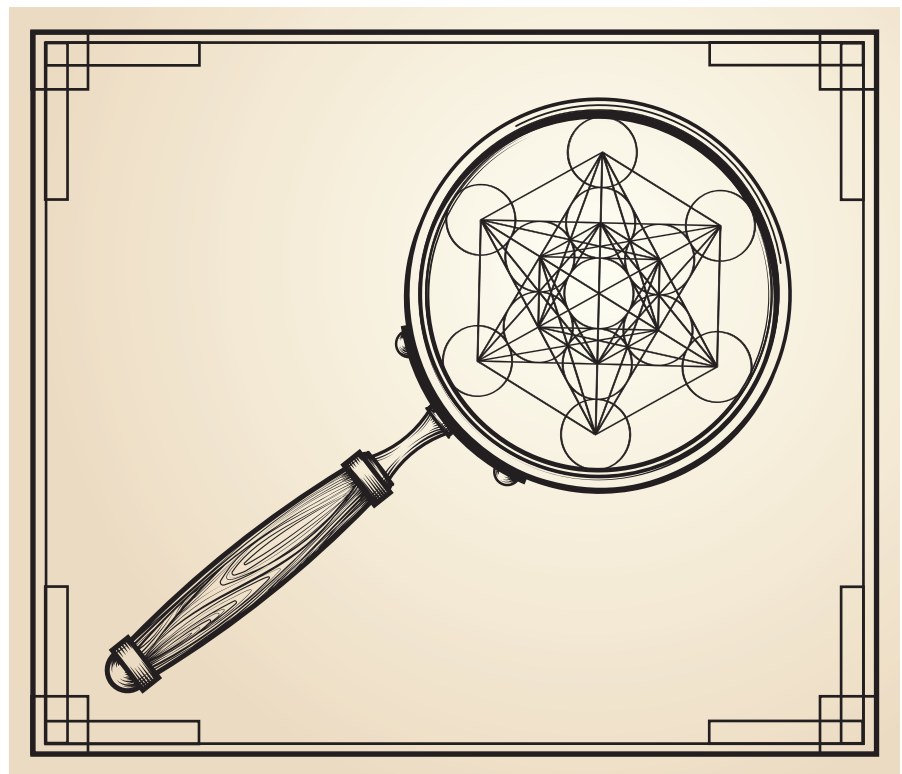
Viewpoint

Using Computer Programs and Search Problems for Teaching Theory of Computation

Recognizing the significance of a cornerstone of computer science.

THE THEORY OF computation is one of the crown jewels of the computer science curriculum. It stretches from the discovery of mathematical problems, such as the halting problem, that cannot be solved by computers, to the most celebrated open problem in computer science today: the P vs. NP question. Since the founding of our discipline by Church and Turing in the 1930s, the theory of computation has addressed some of the most fundamental questions about computers: What does it mean to compute the solution to a problem? Which problems can be solved by computers? Which problems can be solved efficiently, in theory and in practice?

Yet computational theory occupies an ambiguous role in the undergraduate curriculum. It is a required core course for the computer science major at many institutions, whereas at many others it is an upper-level elective. And whether required or not, the theory course can have a reputation as an austere and perhaps even irrelevant niche, disconnected from the skills and ideas that comprise computer science. This is not a new phenomenon, and in recent decades the CS community has worked diligently to improve the accessibility and perceived relevance of the



theory course. Notable contributions include the JFLAP software for experimentation with automata,⁸ and various efforts to promote “NP-completeness for all” via visualizations and practical laboratory exercises.¹

This Viewpoint discusses two specific suggestions for continuing to im-

prove the accessibility of the theory course while maintaining its rigor: first, emphasizing *search problems* rather than decision problems in certain parts of the course; and second, employing *computer programs* written in a real programming language as one of the standard computational

models, complementing the use of automata such as Turing machines. The suggestions here apply specifically to an undergraduate course in which students encounter theory of computation for the first time. The content of such courses varies widely, and the following suggestions are most applicable to introductory courses that incorporate both computability and complexity theory.

Emphasizing Search Problems

The theory of computation is usually phrased in terms of *decision* problems: questions with a single-bit yes/no response. In other areas of computer science, however, we are usually interested in *search* problems, whose solutions consist of more than a single bit. As an example, consider the problem of finding a Hamilton cycle in a graph—that is, a route visiting every vertex exactly once. Theory courses usually discuss the decision problem, “Does the graph G contain a Hamilton cycle?” But if the answer is yes, we still do not know the route of a Hamilton cycle in G . It is more natural and useful to consider the related search problem, “Find and output a Hamilton cycle of G , if one exists.”

Once they have finished their first theory course, computer science un-

dergraduates will recognize the connections between search and decision problems. But search problems are more familiar and more immediately applicable, so there are good reasons to teach the more elementary parts of the theory course with an emphasis on search problems. It is worth noting that Knuth prize winner Oded Goldreich is an advocate of this approach²; his books are among the few modern textbooks^{3,6} that adopt search problems as a primary paradigm. But search problems can easily be incorporated into more traditional approaches that retain decision problems as the standard model, and I do recommend this as a means of connecting the theory course more closely to other parts of the undergraduate CS curriculum.

The advantages of, and techniques for, teaching CS theory via search problems have been discussed in detail elsewhere.⁵ One interesting result, based on a survey of CS undergraduates, is that search problems are perceived as significantly more useful than decision problems. Because perceived relevance is known to be a factor in achieving good learning outcomes, this provides indirect evidence the approach is beneficial.

Using Real Computer Programs to Complement Automata

Another technique for increasing student engagement and connections with other parts of the CS curriculum is to employ code in a real programming language. This can provide a beneficial supplement to the automata and grammars that typically dominate a course in theory of computation. Formal models such as Turing machines are of course essential, especially for providing a rigorous definition of computation itself. However, it is possible to teach a mathematically rigorous theory course using a programming language as the primary model of computation. In this approach, the program model is layered over Turing machines as an underlying model, and Turing machines are still employed when required in certain proofs and definitions. A strong majority of CS theory textbooks do not employ a programming language as the primary computational model, but several authors have done so, for example using Python,⁶ Ruby,⁹ and a variant of LISP.^{4,7} As an example of the approach, consider the Python program shown in the figure here.

This code provides the basis for a proof by contradiction, demonstrating that a certain computational problem is undecidable. Specifically, it proves the undecidability of the following question: “Given a Python function $P()$ and input string I , does P return the value ‘yes’ when invoked with input I ?” A detailed explanation of the proof is outside the scope of this Viewpoint; here, I focus on the potential advantages for undergraduate students who are encountering this type of material for the first time. Note that, in practice, the code shown in the figure would be presented in class only after exposure to and experimentation with prerequisite concepts, such as Python functions that take the source code of other Python functions as input and analyze them or transform them in some way. Nevertheless, for concreteness and compactness in this Viewpoint, I describe the potential benefits to students directly as they appear in this proof.

First, note the undecidability result itself can be described in terms of Python programs: “It is impossible to write a Python program that deter-

Python program example.

```
# yesOnStr(P,I) returns 'yes' if the Python function with
# source code P returns 'yes' after receiving input I.
# We assume yesOnStr(P,I) exists and works correctly
# on all inputs.
from yesOnStr import yesOnStr

# Below is a diagonalized and inverted version of
# yesOnStr(P,I). What happens when the
# parameter P is a string consisting of the
# source code of diagYesOnStr?
def diagYesOnStr(P):
    if yesOnStr(P, P)=='yes':
        return 'no'
    else:
        return 'yes'
```

The source code of Python function `diagYesOnStr()`. This code provides the core of a proof by contradiction. When given its own source code as input, the function `diagYesOnStr()` outputs ‘yes’ if and only if it outputs ‘no’. This contradiction means our assumption that the function `yesOnStr(P, I)` can exist is not valid. Therefore, the problem `YesOnStr` is undecidable: no Python function can correctly answer the question “does Python function P output ‘yes’ on input I ?” for all inputs.

mines whether other Python programs will output 'yes' on a given input." From one point of view, this is a purely cosmetic change from the equivalent statement in terms of Turing machines: "there does not exist a Turing machine that determines whether other Turing machines will accept a given input." After all, students in any theory course must come to understand the equivalence between Turing machines and computer programs. Nevertheless, the practice of discussing results in terms of computer programs that have clear connections to other areas of computer science provides the instructor with opportunities for increased engagement; this has certainly been my own experience.

Second, there are some steps in the theory proofs that are surprisingly subtle when expressed in terms of Turing machines, but become obvious and familiar in a programming language. One example in the figure is the trick in which a single parameter P is duplicated and passed on in two separate roles to the two-parameter function `yesOnStr(P, P)`. In class, this can be further explicated by stepping through the program in a debugger and showing the dual roles of P : as source code in the first parameter and as a text string in the second parameter. (Even Alan Turing recognized the challenges inherent in proofs based on automata. In the seminal 1936 paper that introduced Turing machines, he sympathized with readers who might feel "there must be something wrong" in his first such proof.¹⁰)

Third, students can build an intuitive understanding of code-based proofs by active experimentation with the code. In the example in the figure, one can provide an approximate version of `yesOnStr()` that works correctly on a limited class of inputs. Students can then predict the output of `diagYesOnStr()` on various inputs, and check their answers by running the code. They can construct variants of the code, discussing which variants produce the desired contradiction and which do not. By implementing `yesOnStr()` via simulation, students can discover an important extension to this result: we can in fact write a Python program that always terminates correctly on positive instances of this prob-

Many theory courses could benefit from making more explicit connections to other parts of the computer science curriculum.

lem, so the problem is *recognizable* but not decidable.

Fourth, some students may find the programming approach transfers more easily to novel problems. In recent years I have taught three different approaches for undecidability proofs to all students: traditional reductions employing prose descriptions of Turing machines; explicit Python programs (similar to the example in the figure here) supplemented by a prose explanation of the desired contradiction; and the application of Rice's theorem. In tests and exams, students may choose which proof method to use, and there is an approximately even split among these three proof techniques. In particular, a significant fraction of students choose to write out a Python program as part of their exam answer. This provides empirical evidence that the programming approach is beneficial for some students, and it is plausible all students gain improved understanding from seeing multiple approaches.

Conclusion

Over a period of eight years, I have experimented with techniques for making the undergraduate theory course more accessible and engaging. This Viewpoint suggests two possibilities: emphasizing search problems and employing real computer programs. I do not advocate the universal or complete adoption of these suggestions. I have backed away from some aspects of the approach myself. For example, after experimenting with teaching NP-completeness based on search problems, I concluded this part of the course works better when taught with the traditional focus on decision problems. Similarly, I found there are some technical results

about polynomial-time verifiers that can be proved more instructively—for the target audience of novice undergraduates—using Turing machines rather than computer programs.

Every instructor and every group of students is different; instructors must adopt a style of teaching that is authentic to themselves, achieves the goals of the students, and is based realistically on the students' level of preparedness. I do believe that many theory courses could benefit from making more explicit connections to other parts of the computer science curriculum, and it is possible to do this incrementally. If decision problems and Turing machines are retained as the central paradigms, search problems can be still be mentioned when relevant, and snippets of code can be used to illustrate subtleties.

Whether or not the specific ideas suggested here are adopted, it seems important that we continue to strive for accessibility and engagement in the undergraduate theory course. The theory of computation is a profound and important cornerstone of computer science; I hope that in the years ahead, an ever-growing number of students will appreciate both its beauty and its significant connections to the rest of the computer science curriculum. ■

References

1. Crescenzi, P., Enstrom, E. and Kann, V. From theory to practice: NP-completeness for every CS student. In *Proceedings of ITICSE*, 2013.
2. Goldreich, O. On teaching the basics of complexity theory. *Theoretical Computer Science: Essays in Memory of Shimon Even*. (2006), 348–374.
3. Goldreich, O. *P, NP, and NP-Completeness: The Basics of Computational Complexity*. Cambridge University Press, 2010.
4. Jones, N.D. *Computability and Complexity: From a Programming Perspective*. MIT Press, 1997.
5. MacCormick, J. Strategies for basing the CS theory course on non-decision problems. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*, 2018.
6. MacCormick, J. *What Can Be Computed?: A Practical Guide to the Theory of Computation*. Princeton University Press, 2018.
7. Reus, B. *Limits of Computation: From a Programming Perspective*. Springer, 2016.
8. Rodger, S.H. and Finley, T.W. *JFLAP: An Interactive Formal Languages and Automata Package*. Jones & Bartlett, 2006.
9. Stuart, T. *Understanding Computation: From Simple Machines to Impossible Programs*. O'Reilly, 2013.
10. Turing, A.M. On Computable Numbers, With An Application To The Entscheidungsproblem. In *Proc. London Math Soc.*, Vol. 2–42, 1, (1937), 230–265.

John MacCormick (jmac@dickinson.edu) is Associate Professor of Computer Science at Dickinson College, Carlisle, PA, USA. He is the author of *Nine Algorithms That Changed the Future: The Ingenious Ideas That Drive Today's Computers*.

Copyright held by author.

Article development led by [acmqueue](https://queue.acm.org)
queue.acm.org

Hitting a nerve with field-programmable gate arrays.

BY OSKAR MENCER, DENNIS ALLISON, ELAD BLATT, MARK CUMMINGS, MICHAEL J. FLYNN, JERRY HARRIS, CARL HEWITT, QUINN JACOBSON, MAYSAM LAVASANI, MOHSEN MOAZAMI, HAL MURRAY, MASOUD NIKRAVESH, ANDREAS NOWATZYK, MARK SHAND, AND SHAHRAM SHIRAZI

The History, Status, and Future of FPGAs

THIS ARTICLE IS a summary of a three-hour discussion at Stanford University in September 2019 among the authors. It has been written with combined experiences at and with organizations such as Zilog, Altera, Xilinx, Achronix, Intel, IBM, Stanford, MIT, Berkeley, University of Wisconsin, the Technion, Fairchild, Bell Labs, Bigstream, Google, DIGITAL (DEC), SUN, Nokia, SRI, Hitachi, Silicom, Maxeler Technologies, VMware, Xerox PARC, Cisco, and many others. These organizations are not responsible for the content, but may have inspired the authors in some ways, to arrive at the colorful ride through FPGA space described here.

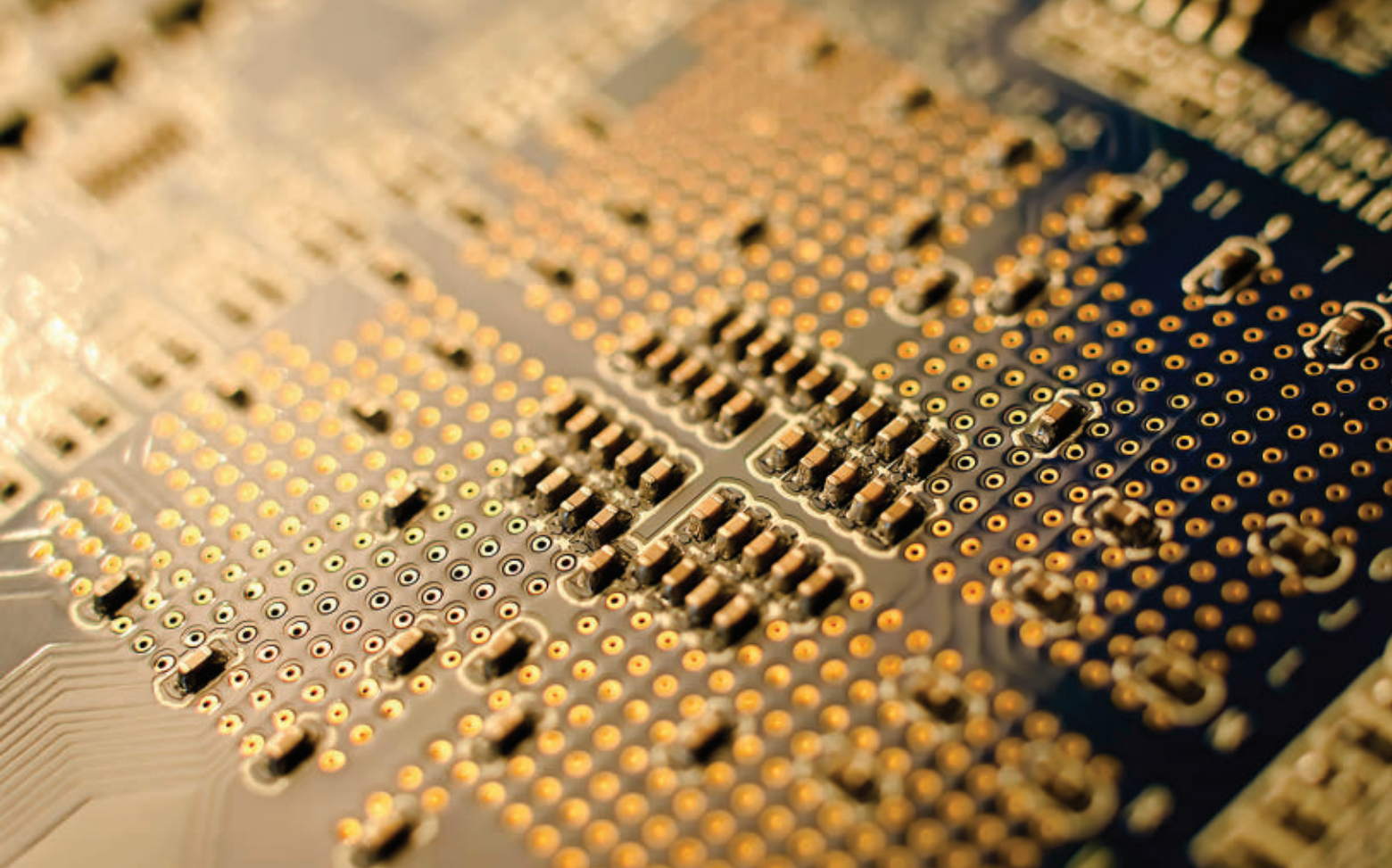
Field-programmable gate arrays (FPGAs) have been hitting a nerve in the ASIC community since their inception. In the mid-1980s, Ross Freeman and his colleagues bought the technology from Zilog and started Xilinx, targeting the ASIC emulation and education markets. (Zilog came out of Exxon, since in the 1970s people were already afraid that oil would run out in 30 years, which is still true today). In parallel, Altera was founded with similar technology at its core.

An FPGA is a chip that is programmed by a circuit. It is said to “emulate” that circuit. This emulation runs slower than the actual circuit would run if it were implemented in an ASIC—it has a slower clock frequency and uses more power, *but* it can be reprogrammed every few hundred milliseconds.

People who make ASICs started using FPGAs to emulate their ASICs before committing them to a mask and sending them out to the factory to be manufactured. Intel, AMD, and many other companies use FPGAs to emulate their chips before manufacturing them.

The telecom industry has been a heavy user of FPGAs. Telecom standards keep changing and building telecom equipment is difficult, so the company that ships telecom solutions first tends to capture the biggest chunk of the market. Since ASICs take a long time to make, FPGAs offer an opportunity for a shortcut. FPGAs started to be adopted for first versions of telecom equipment, which initiated the FPGA price conflict. While the price of the FPGA does not matter to the ASIC emulation market, the price of a chip for telecom is important. Many years ago, AT&T and Lucent made their own FPGAs, called ORCAs (optimized reconfigurable cell arrays), but they were not competitive with Xilinx or Altera in terms of speed or size of the silicon.

Today, Huawei is the largest customer for FPGAs. It is possible the recent tension between the U.S. and China began with FPGAs from the U.S. giving Huawei an edge in delivering 5G telecom equipment two years before any of



the other vendors around the world got ready to play.

FPGA price hits a nerve. Early on, FPGAs were used for software-defined radios (SDRs), building radios for communication on many different standards at the same time, in essence having a single phone speaking many languages. This time, FPGAs hit a huge nerve. There was a split in how SDR technology was implemented. Commercial vendors developed cost-effective solutions, and today every base station on the planet has SDR technology in it. In the defense community, on the other hand, SDRs were built by large defense contractors with profitable legacy product lines to protect. The result was that the price of FPGA-based radio products was so high that a part of the U.S. defense market got a persistent allergic reaction to their use.

Next, FPGAs tried to grow in the DSP (digital signal processor) and embedded markets. FPGAs with little hard microprocessors in the corner started to appear. The pressure to sell these new FPGAs was so high that if customers rejected the new family of chips, they were put on a blacklist, and sometimes even refused service for a few months. Pressure to grow the FPGA market was and

still is immense, as is the magnitude of the failures of FPGA companies to conquer new markets, given the impossibility of reducing the price of FPGA products because of their enormous surface area and layers of intellectual property.

Hitting a nerve in HPC and datacenters. For the past few years, FPGAs have tried to grow in the high-performance computing (HPC) and datacenter markets. In 2017, Microsoft announced its use of Altera FPGAs in the datacenter, and Intel bought Altera. In 2018, Xilinx announced its “Datacenter First” strategy, with the Xilinx CEO declaring in front of an audience of analysts that Xilinx is not an FPGA company *anymore*. This may have been a slight dramatization, but historically there is relevance.

In HPC and datacenter usage of FPGAs, the main obstacle today is *place and route*—the time it takes to run the proprietary FPGA vendor software that maps the circuit onto the FPGA elements. On large FPGAs and on a fast CPU server, place and route takes up to three days, and many times even after three days the software fails to find a mapping.

Hitting a nerve in oil and gas. In oil and gas implementations, however, around 2007 a niche opened up. The

time it took classical computers to simulate the drilling of holes in the earth to find oil was longer than the actual building of a drilling site and the drilling itself. The use of FPGA accelerators dramatically changed this upside-down timing. The first FPGAs in the datacenter of an oil company, computing seismic images, were built by Maxeler Technologies and delivered to Chevron.³

The use of FPGAs in oil and gas expanded for a few years, until pressure from the ASIC industry led to a return to standard CPU technology. Today prediction and simulations in oil and gas are still important, and seismic imaging is mostly done on CPUs and GPUs, but the FPGA opportunity still exists. We are reminded that “today’s new stuff is tomorrow’s legacy,” and, of course, today’s new stuff is AI and a focus on data.

Despite all of this, FPGAs remain a quick way to market, a simple way to obtain competitive advantage, and an indispensable technology for many mission-critical situations—even though they are expensive on a per-chip basis compared with ASICs. In HPC and the datacenter, however, FPGAs have significantly lower operational costs compared with running soft-

ware on CPUs or GPUs. Fewer FPGAs are needed, requiring much less cooling than both CPUs and GPUs. FPGAs make for smaller datacenters, hitting a nerve with operators who fear their datacenters might shrink.

ASIC vs. FPGA. Another way to use FPGAs is to complement ASICs. ASICs are built to hold fixed functionality while adding FPGAs to provide some flexibility for last-minute changes or adaptivity of the products to different markets.

Modern FPGAs are integrating more hard functionality and becoming more and more like ASICs—while ASICs are sometimes adding a bit of FPGA fabric into their design for debugging, testing, in-field fixes, and flexibility in adding little bits of functionality as needed.


Nevertheless, ASIC teams always fight the FPGA concept. ASIC designers ask, “Which functionality do you want?” and are impatient if the answer is, “I don’t know yet.”

One such new battleground is the autonomous car industry. Since algorithms are constantly changing, and laws could change when cars are in the field, requiring driver updates, the solution needs to be flexible. FPGAs have a lower clock frequency, and thus smaller heat sinks, resulting in a smaller physical size than CPUs and GPUs. Lower power consumption and smaller size makes FPGAs the obvious choice. Nevertheless, GPUs are easier to program and do not require a three-day place and route.


Moreover, it is critical to be able to run the same code in the car and in the cloud (primarily for simulation and testing), so FPGAs would have to be available in the cloud before they could be used in the car. For these reasons, many developers prefer GPUs.

Evolution of FPGAS

FPGAs are evolving. Modern interfaces are trying to make FPGAs easier to program, more modular, and more cooperative with other technologies. FPGAs support Advanced Extensible Interface (AXI) buses, which make them easier to program but also introduce enormous inefficiencies and make FPGAs less performant and ultimately much less competitive. Academic work, such as Eric Chung’s paper on dynamic networks for FPGAs,¹ helps with the routing prob-



ASIC teams always fight the FPGA concept. ASIC designers ask, “Which functionality do you want?” and are impatient if the answer is, “I don’t know yet.”



lem, but such advanced ideas have not yet been picked up by industry.

How are FPGAs connected? For HPC workloads with large flows of data, you can use PCI Express and deploy communication-hiding techniques. But how about small workloads, such as found in network function virtualization (NFV), serving a large number of users at the same time. For NFV and acceleration of virtual machines in general, the FPGA must connect directly to the CPU, possibly using cache coherency as a communication mechanism, as investigated these days by VMware. Of course, a key feature is the ability to crash the FPGA without crashing the CPU, and vice versa. Hyperscalar technology companies are rediscovering requirements from IBM mainframe days, driving more and more complexity into standardized platforms.

There are also opportunities for the masses. In offering FPGA platforms, organizations without the budgets for ASIC development and without knowledge of the latest silicon fabrication challenges and solutions can develop circuits and build competitive advantage into their products, such as the newly emerging opportunities for computing at the edge of the Internet of Things (IoT) network, close to sensors, displays, or just in-line at the wire, as data flows through.

Meanwhile, FPGA companies are pushing vertically up the stack and into the CPU socket, where Intel is dominating the market, including, for example, special instructions for NFV. The key barriers to entry for new CPUs and FPGAs in the datacenter are not just speed and cost, but also the availability of software and drivers for all possible I/O devices.

Key to making FPGAs work in the datacenter is to make them easier to use—for example, with automatic tools that drive the use of FPGAs without place and route difficulties. Microsoft pioneered the use of FPGAs in a hyperscalar datacenter for accelerating Bing, NFV, and AI algorithms. Microsoft also built abstractions, domain-specific languages, and flexible hardware infrastructures. Commercially, the main problem with FPGAs is the go-to-market strategy.

Building new chips and *then* starting to think about the software is too late.

How do you extract value from existing software by adapting the hardware to serve the software? This also brings an opportunity to rethink FPGA architecture. A word of warning, however: The silicon industry devours cash. Building ASICs is a poker game with minimum bets rising over the years. It's a winner-take-all game, and any threats such as FPGAs get eliminated early in the race.

FPGAs are creating additional and undesirable risks for silicon projects.

Niche Technology

While a software designer will always say, "If it *can* be done in software, it *will* be done in software," the ASIC designer will say, "If it *can* be done in an ASIC, it *will* be done in an ASIC." Most interestingly, "If it can be done in software, you don't have to deal with the guy who thinks like an FPGA." FPGAs have a tiny community of many, sometimes eccentric, programmers, compared with the armies needed to make ASICs and with the world population of software programmers. The FPGA companies are small. The FPGA community is small.

Intel is driving FPGAs for flexibility. It is the most successful company following the principle of building the hardware to run existing software.

FPGAs can be faster than CPUs and GPUs, but the hard lesson from industry and the investment community is that most of the time during a computer's existence, speed does not matter, and real time does not matter. Therefore, buying a computer for speed alone is rare. It happens, but it's more of a random event than a market on which to build a business. In addition, FPGAs have no standard, open source, enjoyable programming model—and, therefore, no standard marketplace for FPGA programs that work on all FPGA chips or can be easily cross-compiled. Maxeler Technologies has a high-level solution to provide such an interface, but wide industry adoption requires trust. To go from early adopters to benefiting everyone, trust requires alignment and support from established vendors in the datacenter space.

Applications people in the real world say, "I don't care what it is, just give me a way to do what I want to do." What are the possible application areas for FPGAs that have not been widely explored yet? For real-time computing, there is

manufacturing. For computer vision on drones, it's the weight and power advantage of FPGAs. On a satellite it is very expensive to do hardware upgrades, so FPGAs provide long-term flexibility that can be critical. FPGAs need to find a product that resonates, and they need to be easy to program. It's not just the hardware or software, it's the ecosystem. It's the complete solution.

One way to expand beyond current market confines is real-time compilation and automatic FPGA program generation. This is easier said than done, but the opportunity is growing with AI tearing up the application space. These days, everything is done with AI; even traditional algorithms such as seismic imaging for oil and gas are incorporating AI. A science and engineering solution is needed to deal with AI blocks. FPGAs might be a good starting point, maybe initially to connect the AI blocks and then to incorporate them into the FPGA fabric such as the next-generation chips from Xilinx—with AI fabric, CPUs, 100G interfaces, and FPGA cells all in the same 7-nm chip.

From another perspective, with AI chips producing and consuming vast amounts of data, FPGAs will be needed to feed the beast and move outputs away swiftly. With all the new ASICs for AI processing coming out, FPGAs could provide differentiation to AI chip companies.

Predictions

Could the following developments have been predicted 10 or 25 years ago?² While the world changes, the predictions seem to stay the same.

1. There will be successful CPU+FPGA server chips, or FPGAs with direct access to the CPU's cache hierarchy. Some say yes, and some say no.

2. System on a chip (SoC) FPGA chips will grow and expand, driving the medical, next-generation telecom, and automotive industries, among others.

3. Developers will use FPGAs to do amazing things and make the world a better place but will have to hide the fact that there is an FPGA inside.

4. The FPGA name will remain, and chips called FPGAs will be built, but everything inside will be completely different.

5. As we forego (dataflow) optimization in order to make FPGAs easier to

program, the performance of FPGAs will be reduced so they are no longer competitive with CPUs, which will always be easier to program.

6. There will be FPGAs with dynamic routing, evolving interconnect, and run-time-flexible data movement.

7. Place and route software, as well as the complete software stack on top of FPGAs, will be open source. There are already initial efforts with Yosys and Lattice FPGAs.

8. All semiconductor architectures will be combined into single chips with combinations of TPUs, GPUs, CPUs, ASICs, and FPGAs. Some may be combinations of the whole of each. Others will be combinations of parts of each.

9. More chips will be focused on limited application spaces, and fewer on general-purpose chips. In a way, everything is becoming an SoC.

Final Comment

How many conflicts are resolved with this article, and how many new ones are created? In this sense, a conflict is a challenge to an existing way of doing things. Such an existing way of doing things may have implications for the way people think, and, therefore, for the way they act. But maybe more importantly, there will be implications on how we developers earn a living. ■

Related articles on queue.acm.org

FPGA Programming for the Masses

David F. Bacon, Rodric Rabbah, Sunil Shukla
<https://queue.acm.org/detail.cfm?id=2443836>

FPGAs in Data Centers

Gustavo Alonso
<https://queue.acm.org/detail.cfm?id=3231573>

Reconfigurable Future

Mark Horowitz
<https://queue.acm.org/detail.cfm?id=1388771>

References

1. Chung, E. CoRAM: An in-fabric memory architecture for FPGA-based computing. Ph.D. thesis, 2011. Carnegie Mellon University, Pittsburgh, PA, USA.
2. Field-programmable Custom Computing Machines. FCCM predictions, 2012; <https://www.fccm.org/past/2012/Previous.html>.
3. Nemeth, T., Stefani, J., Liu, W., Dimond, R., Pell, O., Ergas, R. An implementation of the acoustic wave equation. In *Proceedings of the 78th Society of Exploration Geophysicists Meeting*, (Las Vegas, NV, 2008).

Contact **Oskar Mencer** (mencer@maxeler.com) with any inquiries about this article.

Copyright held by authors/owners.
Publication rights licensed to ACM.

Article development led by [acmqueue](https://queue.acm.org)
queue.acm.org

How experts debug production issues in complex distributed systems.

BY CHARISMA CHAN AND BETH COOPER

Debugging Incidents in Google's Distributed Systems

GOOGLE HAS PUBLISHED two books about Site Reliability Engineering (SRE) principles, best practices, and practical applications.^{1,2} In the heat of the moment when handling a production incident, however, a team's actual response and debugging approaches often differ from ideal best practices.

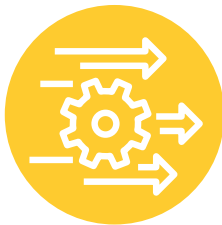
This article covers the outcomes of research performed in 2019 on how engineers at Google debug production issues, including the types of tools, high-level strategies, and low-level tasks that engineers use in varying combinations to debug effectively. It examines the research approach used to capture data, summarizing the common engineering journeys for

production investigations and sharing examples of how experts debug complex distributed systems. Finally, the article extends the Google specifics of this research to provide some practical strategies that you can apply in your organization.

As this study began, its focus was on developing an empirical understanding of the debugging process, with the overarching goal of creating optimal product solutions that met the needs of Google engineers. We wanted to capture the data that engineers need when debugging, when they need it, the communication process among the teams involved, and the types of mitigations that are successful. The hypothesis was that commonalities exist across the types of questions that engineers are trying to answer while debugging pro-

Mitigate early

Establish SLOs and accurate monitoring



Triage effectively



Apply established strategies for common issues



duction incidents, as well as the mitigation strategies they apply.

To this end, we analyzed postmortem results over the last year and extracted time to mitigation, root causes, and correlated mitigations for each. We then selected 20 recent incidents for qualitative user studies. This approach allowed us to understand and evaluate the processes and practices of engineers in a real-world setting and to deep-dive into user behavior and patterns that couldn't be extracted by analyzing trends in postmortem documents.

The first step was trying to understand user behavior: At the highest level, what did the end-to-end debugging experience look like at Google? The study was broken down into the following phases (which are unpacked in the sections that follow):

► **Phase 0:** Define a way to segment the *incident responder* and *incident type* populations.

► **Phase 1:** Audit the postmortem documentation from a spread of actual Google incidents.

► **Phase 2:** Conduct in-depth user interviews with first responders who worked on those incidents.

► **Phase 3:** Map the responders' journeys across those incidents, detailing common patterns, questions, and steps taken.

Phase 0: Segment incident responder and incident type populations. The preliminary approach to segmenting the population under study was designed to ensure a sufficiently broad set of incidents and interviewees was included, from which we could capture a comprehensive set of data.

Incident responders. First, the incident responders (or on-callers) were segmented into two distinct groups: SWEs (software engineers), who typically work with a product team, and SREs (Site Reliability Engineers), who are often responsible for the reliability of many products. These two groups were further segmented according to tenure at Google. We found the following behaviors across the different user cohorts:

SWE vs. SRE mental models and tools. SWEs are more likely to consult logs earlier in their debugging workflow, where they look for errors that could indicate where a failure occurred.

SREs rely on a more generic approach to debugging: Because SREs are often on call for multiple services, they apply a general approach to debugging

based on known characteristics of their system(s). They look for common failure patterns across service health metrics (for example, errors and latency for requests) to isolate where the issue is happening, and often dig into logs only if they're still uncertain about the best mitigation strategy.

Experience level of the incident responder. Newer engineers are more likely to use recently developed tools, while engineers with extensive experience (10 or more years running com-

plex, distributed systems at Google) tend to use more legacy tools. Intuitively, this finding makes sense—people tend to use the tools they are most comfortable with, particularly in emergency situations.

Incident types. We also examined incidents across the following dimensions, and found some common patterns for each:

► **Scale and complexity.** The larger the blast radius (that is, its location(s), the affected systems, the importance of the

user journey affected, and so on) of the problem, the more complex the issue.

► **Size of the responding team.** As more people are involved in an investigation, communication channels among teams grow, and tighter collaboration and handoffs between teams become even more critical.

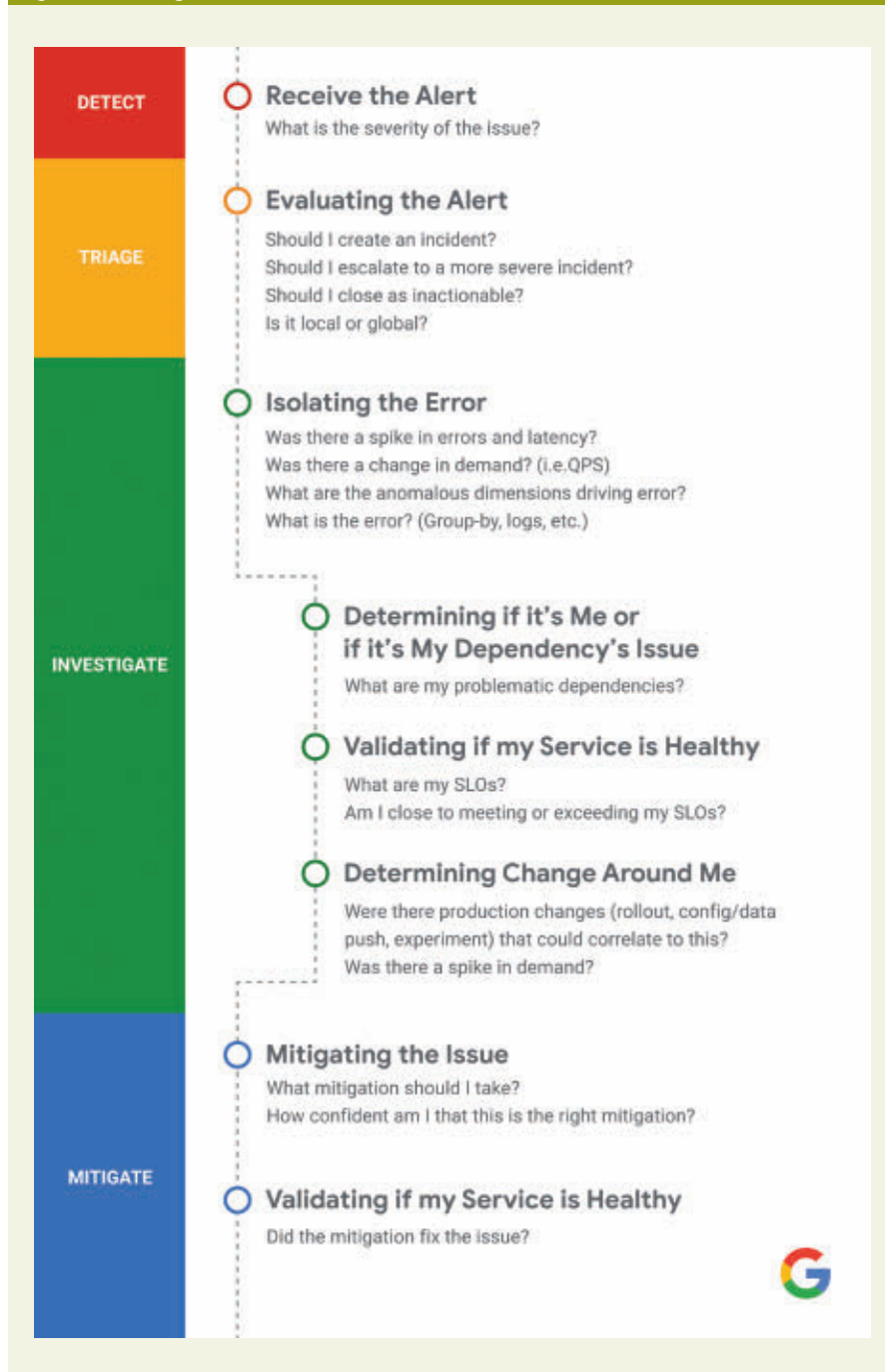
► **Underlying cause.** On-callers are likely to respond to symptoms that map to six common underlying issues: capacity problems; code changes; configuration changes; dependency issues (a system/service my system/service depends on is broken); underlying infrastructure issues (network or servers are down); and external traffic issues. Our investigation intentionally did not look at security or data-correctness issues since they were outside the scope of the tools focused on in this work.

► **Detection.** On-callers learn about issues through human or machine detection that is based on availability or performance problems. Some common mechanisms include alerts on the following: white-box metrics; synthetic traffic; SLO (service-level objective) violations; and user-detected issues.

Phase 1: Postmortem documentation analysis. Once the different categories of incidents were determined, we read the postmortems for the 20 incidents identified for qualitative studies, mapping the steps responders took in each case. This approach allowed us to validate the common factors that affect how responders handled these incidents and the challenges they faced. We could also ensure that the incidents selected for deep-dive analysis were distributed across the dimensions, as just described.

Google has a strong culture of blameless postmortems.⁴ It is common for teams to look at the history of their failures to ensure that their services are continuing to run reliably. Because of this, postmortem documents are readily available internally and were an invaluable resource for analyzing debugging behavior. Detailed chat transcripts linked to these postmortems helped form a base understanding of what happened, when it happened, and what went wrong. We could then start mapping a prototype of the debugging journey. Future research could extend this work by ap-

Figure 1. Building blocks.



plying natural-language processing to further validate response patterns in the incident response chats.

Phase 2: In-depth interviews. To round out this study, in-depth interviews were conducted with the first responders identified in these 20 post-mortems so any gaps in the postmortem document could be filled in. These data sources added significant color to the debugging journey we were mapping, and surfaced a core set of building blocks that make up the overall debugging process.

Phase 3: Mapping the responders' journeys. This study allowed us to generate snapshots of what an actual incident investigation life cycle looks like at Google. By mapping out each responder's journey and then aggregating those views, we extracted common patterns, tools, and questions asked around debugging that apply to virtually every type of incident. Figure 1 is a sample of the visual mapping of the steps taken by each of the responders interviewed.

Common Patterns around Debugging

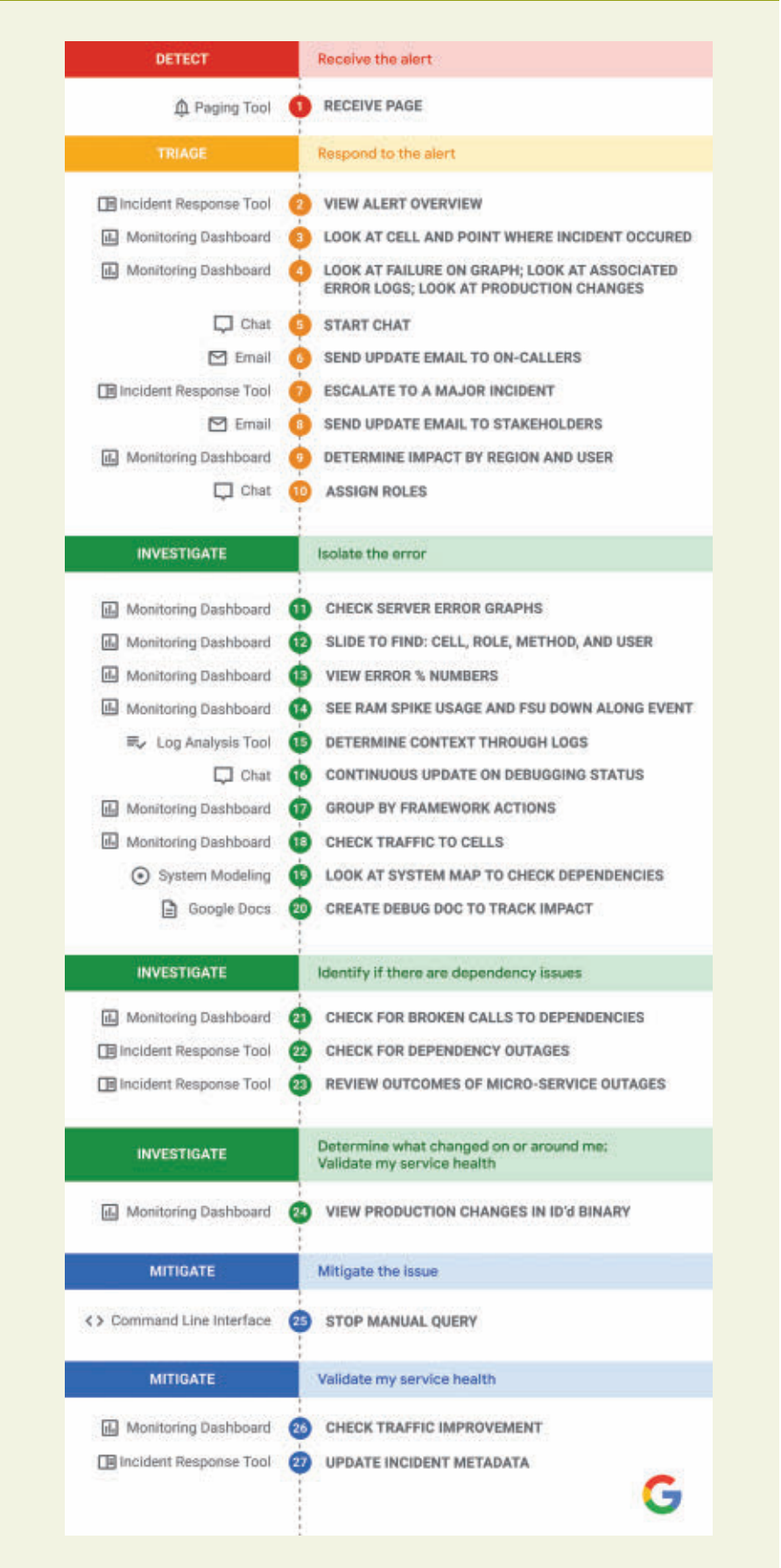
A typical canonical debugging journey consists of the stages and sub-journeys shown in Figure 2 and described here. These building blocks are often repeated as the user investigates the issue, and each block can happen in a nonsequential and, sometimes, cyclical order.

During the detection to mitigation stages, investigations are typically time sensitive—especially when the issue affects the end-user experience. An on-caller will always try to mitigate the issue or “stop the bleeding” before uncovering the root cause. After mitigation, on-callers and developers often perform a deeper analysis of the code and apply measures to prevent a similar situation from recurring.

Detect. The on-caller discovers the issue via an alert, a customer escalation, or a proactive investigation by an engineer on the team. A common question would be: What is the severity of this issue?

Triage loop. The on-caller's goal is to assess the situation quickly by examining the situation's blast radius (the severity and impact of the issue) and determining whether there is a need to escalate (pull in other teams, inform

Figure 2. User journey.



internal and external stakeholders). This stage can happen multiple times in a single incident as more information comes in.

Common questions include: Should I escalate? Do I need to address this issue immediately, or can this wait? Is this outage local, regional, or global? If the outage is local or regional, could it become global (for example, a rollout contained by a canary analysis tool likely won't trigger a global outage, whereas a query of death triggered by a rollout that is now spreading across your systems might)?

Investigate loop. The on-caller forms hypotheses about potential issues and gathers data using a variety of monitoring tools to validate or disprove theories. The on-caller then attempts to mitigate or fix the underlying problem. This stage typically happens multiple times in a single incident as the on-caller collects data to validate or disprove any hypotheses about what caused the issue.

Common questions include: Was there a spike in errors and latency? Was there a change in demand? How unhealthy is this service? (Is this a false alarm, or are customers still experiencing issues?) What are the problematic dependencies? Were there production changes in services or dependencies?


Mitigate loop. The on-caller's goal is to determine what mitigation action could fix the issue. Sometimes a mitigation attempt can make the issue worse or cause an adverse ripple effect on one of its dependent services. Remediation (or full resolution of the issue) usually takes the longest of all the debugging steps. This step can, and often does, happen multiple times in a single incident.

Common questions include: What mitigation should be taken? How confident are you that this is the appropriate mitigation? Did this mitigation fix the issue?

Resolve/root-cause loop. The on-caller's goal is to figure out the underlying issue in order to prevent the problem from occurring again. This step typically occurs after the issue is mitigated and is no longer time sensitive, and it may involve significant code changes. Responders write the postmortem documentation during this stage.



Sometimes a mitigation attempt can make the issue worse or cause an adverse ripple effect on one of its dependent services.



Common questions include: What went wrong? What's the root cause of the problem? How can you make your processes and systems more resilient?

Communication. Throughout the entire process, incident responders document their findings, work with teammates on debugging, and communicate outside of their team as needed.

Observability Data

In every single interview, on-callers reported that they started working with time-series metrics that indicate the health of a given service, performing a breadth-first search to identify which components of the system were broken. The majority of the teams that were interviewed evaluated the following items:

- ▶ RPC (remote procedure call) latency and error metrics (similar to the metrics derived from the open source gRPC libraries).
- ▶ Change in external traffic, including QPS (queries per second).
- ▶ Change in production such as rollouts, configuration pushes, and experiments.
- ▶ Underlying job metrics such as memory and CPU consumption.

Both alerts and real-time dashboards use these metrics. On-callers typically used logs and traces only after they identified a component as broken, and they then needed to drill down to the specific issue.

Anecdotes from the Front Line

Some of the interviewees applied SRE best practices to debug complex distributed systems, methodically eliminating their theories on what could go wrong, applying temporary mitigations to prevent user pain, and, finally, successfully resolving and root-causing the problem that set off the outage in the first place.

Many other responders hit unexpected roadblocks. Some responders were impacted by a complex set of changes throughout the stack that occurred simultaneously. Therefore, it was extremely challenging to isolate the actual issue and figure out how to resolve it. Other responders cited process and awareness issues: Some did not fully understand how their production tooling worked, or the appropriate standard course of action to take. Some

responders wound up unintentionally applying bad changes to production.

Following are some (anonymous) stories to illustrate successful and problematic debugging sessions. These anecdotes are intended to show that even with the most experienced engineers, great technology, and powerful tooling, things can—and do—go wrong in unexpected ways.

An exemplary debugging journey.

The following is an example of a successful debugging session, where the SRE follows best practices and mitigates a service-critical issue in less than 20 minutes.

While sitting in a meeting, the SRE on-caller receives a page informing her the front-end server is seeing a 500-server error. While she is initially looking at service health dashboards, a pager-storm starts, and she sees many more alerts firing and errors surfacing. She responds quickly and immediately identifies that her service isn't healthy.

She then determines the severity of the issue, first asking herself how many users are affected. After looking at a few error rate charts, she confirms that a few locations have been hit with this outage, and she suspects that it will significantly worsen if not immediately addressed. This line of questioning is referred to as the *triage loop*, similar to triage processes used in health care (for example, emergency rooms that sort patients by urgency or type of service). The SRE needs to determine if the alert is noise, if she needs to handle it now, and whether to escalate the issue to other teams and stakeholders.

Now that she knows this is a real and relatively severe issue, the SRE starts pulling in other people from her team to help with the investigation. She also sets up communication channels to inform other teams that may be affected, and to let them know her team is addressing the outage.

She then focuses on temporarily mitigating the issue for end users. She tasks a teammate with ensuring traffic isn't routed to any of the unhealthy locations and configuring load balancers to avoid sending traffic to affected locations. For the moment, this action stops the issue from propagating, which leaves her free to conduct a deeper investigation using monitoring data.

Next, she asks a series of questions that help her narrow down the potential cause and figure out how best to mitigate the issue permanently. She largely uses time-series metrics (for example, cloud monitoring metrics³) to help answer these questions quickly:

- ▶ *To narrow down the breadth of the investigation:* Which specific parts of the service are unhealthy? Are the errors coming from the front end or the back end? Are there “slices” of data that are problematic? Are there outliers in the data?

- ▶ *To identify the severity of the issue and rule out causes:* Is the shape of the graph a step (something changed suddenly and remained unchanged), a spike (something changed, then stopped), or a slope (a gradual rollout is happening)? How quickly did the error rate ramp up?

- ▶ *To identify the severity:* What is the blast radius? (If errors occur globally, this indicates a severe issue that will most likely have end-user impact.)

- ▶ *To rule out underlying causes:* When did the problem start? What production events in the service or in its dependencies correlate with this issue?

Once the issue is mitigated, the SRE drills into logs and traces, confirming that a new line of code was crashing the jobs in the regions with issues. She decides to roll back to the last stable version of the service, and validates that the issue is resolved when the affected locations are brought back online.

Debugging journey where the tooling failed to support the on-caller. The following is an example of a journey where Google on-callers hit unexpected hurdles as they debugged, and where applying best practices could have reduced the time to mitigation.

The on-caller receives a page that informs him that the service's overall server-side availability SLO (service-level objective) was down from 99.9% to 91%, and that specific user actions failed. He begins his investigation by looking at graphs of metrics that confirm when the error rate started to increase; errors were mostly caused by timeouts; and, request durations were about equal to the duration of the timeout. He then slices the metrics to the failing user actions identified before, checks the associated server er-

rors and queries-per-second metrics, and digs into server logs to find specific errors. Up to this point, he has followed common practices for debugging.

At the same time, another on-caller for a back-end service dependency notices the service is nearing its quota limitations and suspects that this situation might have an impact on the investigation. This on-caller tries to allocate some quota through a configuration change, hoping to alleviate the problem. Because of a misunderstanding in the configuration push tooling, however, this change accidentally removes a back-end server in one location instead of adding quota, which increases the error rates in the other locations. Additionally, since he considered this change to be safe, the on-caller did not monitor the rollout of the updated configuration as closely as best practices recommend, and initially missed indicators that overall capacity was actually reduced by removing that location. At this point, the on-caller breaks from best practices by performing a global push of a nonvalidated configuration that includes a completely unrelated change—the action of dropping a back end should be separate from adding capacity.

While this is happening, the first on-caller goes deep in the logs and finds “permission-denied” errors increased at the time the back-end server was removed. He does this through a breadth-first search of a number of the supporting back ends and an analysis of their aggregated logs. Here, he notices that when one server was removed, more requests were funneling to the servers that were experiencing issues. Only after digging into logs and opening a number of tools is he able to connect the errors to the configuration change in the dependency.

Better tooling could have prevented the user from performing an unanticipated change. Tooling could also have helped validate what the change would actually do. Additionally, better tooling to support monitoring the effects of the changes to the system could have helped the on-callers draw these conclusions earlier.

The on-callers then connect to share their findings. Once connected, the

first on-caller rolls back the configuration push that reduced capacity, identifies the back-end dependency that changed the permission errors, and works with the back-end team to get bad changes rolled back.

Translating Insights into Concrete Action

If you are responsible for running a distributed service, you might find yourself dealing with scenarios similar to what the teams we interviewed experienced. Our study revealed teams that apply the following principles are typically able to mitigate service problems faster.

Establish SLOs and accurate monitoring. You need to have SLOs and/or metrics that you can alert and optionally report on. These should accurately reflect user pain and allow for slicing by failure domains. These should also be associated with alerts that have clear next steps and links to the most important information.

Triage effectively. Once you have the prerequisites of SLOs and accurate monitoring in place, you need to be able to quickly determine both the severity of user pain and the total blast radius. You should also know how to set up the proper communication channels based on the severity of the issue.

Mitigate early. Documenting a set of mitigation strategies that are safe for your service can help on-callers temporarily fix user-facing issues and buy your team critical time to identify the root cause. For more information on implementing generic mitigations (see Mace.⁵) The ability to easily identify what changed in your service—either in its critical dependencies or in your user traffic—is also helpful in determining what mitigation attempt to move forward with. As mentioned in the exemplary debugging case, asking a series of common questions and having metrics, logs, and traces can help speed up the process of validating your theories about what went wrong.

Apply established mitigation strategies for common issues. Although every service is different, the following patterns emerged in the underlying issues we examined and the mitigations associated with them. When you are

dealing with a problem that you have never seen before, it can be helpful to think about what type of issue your service is facing, the questions you should ask, and the associated mitigations based on the answers.

► *Service errors.* This was the most common cause for an alert firing in our study. As such, it also had the largest variety of mitigations. Some factors to consider in determining mitigation strategies include: Are the errors occurring globally? Check for correlated rollouts, configuration/data changes, and experiments. Are the incoming QPS spiking? Add capacity and/or start load shedding to drop traffic that your service can't handle. Is a bad actor causing a change in QPS? If so, block the user.

► *Performance.* Latency can make for a bad user experience and degrade into errors over time. These issues can be difficult to debug if there is no obvious correlated capacity or production change. Typically, responders look through traces to identify which components in the stack are affected and try to determine a solution from there.

► *Capacity.* Capacity issues are some of the easiest to spot, especially if you have capacity-specific alerts. Like errors and performance issues, these can manifest as both fast and slow burns. If a service is going to run out of capacity immediately, teams typically ask for more capacity in an “emergency loan” to scale up their service (or they may attempt to scale out). For a slow burn, responders perform additional analyses and planning to determine if there are other underlying issues. These types of alerts surface only when automated capacity systems hit their authorized maximum, and acquiring more resources requires human intervention.

► *Dependency issues.* A critical dependency—even if it is deep within the service stack—can contribute to the failure of the entire service. Knowing your hard dependencies (those in the critical path of your code) and being able to view the health of these dependencies can be helpful in ruling out whether the problem actually lies with another service.

► *Debugging microservices.* Most of the teams we interviewed have a microser-

vice architecture. Frequently, the error may be deeper in the stack than where it manifested to the on-caller. Similar to debugging dependencies, it's helpful to be able to traverse the stack quickly, associate production changes, and understand service architecture.

Conclusion

SREs continuously strive to improve systems and expose vulnerabilities in order to limit the probability of failures, near misses, and inefficiencies in production. Even under the most ideal conditions, things inevitably go wrong. By surfacing, preserving, and disseminating the commonalities—both positive and negative—in the debugging workflow, the aim is to prevent the same class of problem from recurring, or, when prevention isn't possible, to minimize the duration or impact of unavoidable outages. Hopefully, other organizations can apply these findings in practice too. □

Related articles on queue.acm.org

The Calculus of Service Availability

Ben Treynor, Mike Dahlin, Vivek Rau, Betsy Beyer
<https://queue.acm.org/detail.cfm?id=3096459>

Why SRE Documents Matter

Shylaja Nukala and Vivek Rau
<https://queue.acm.org/detail.cfm?id=3283589>

Weathering the Unexpected

Kripa Krishnan
<https://queue.acm.org/detail.cfm?id=2371516>

References

1. Beyer, B., Jones, C., Petoff, J., and Murphy, N.R., Eds. *Building Secure and Reliable Systems*. O'Reilly Media, 2016; <https://landing.google.com/sre/books/>.
2. Beyer, B., Murphy, N.R., Rensin, D.K., Kawahara, K., and Thorne, S., Eds. *The Site Reliability Workbook: Practical Ways to Implement SRE*. O'Reilly Media, 2018; <https://landing.google.com/sre/books/>.
3. Google Cloud. Metric list, 2020; <https://cloud.google.com/monitoring/api/metrics>.
4. Lunney, J. and Lueder, S. *Postmortem Culture: Learning From Failure*. O'Reilly Media, 2017; <https://landing.google.com/sre/sre-book/chapters/postmortem-culture/>.
5. Mace, J. *Spotlight on Cloud: Reducing the Impact of Service Outages with Generic Mitigations with Jennifer Mace*. O'Reilly Media, 2019; <https://www.oreilly.com/library/view/spotlight-on-cloud/0636920347927/>.

Charisma Chan is a user experience design researcher at Google U.K. in London. Prior to joining Google, she led design research and strategy for consumer and enterprise products in the financial services and media sectors.

Beth Cooper is a product manager at Google in New York City. She focuses on building Google scale monitoring for both site reliability and software engineers. Prior to Google, she worked on Microsoft Azure building products for cloud and datacenter automation.

Copyright held by authors/owners.
 Publication rights licensed to ACM.

IoTDI 2021

6th ACM/IEEE International Conference on Internet of Things Design and Implementation

May 18-21, 2021 – CPS-IoT Week – Nashville, Tennessee, USA
<https://conferences.computer.org/iotDI/2021/>

ACM/IEEE IoTDI is the premier venue for all topics related to the Internet of Things. The conference is an interdisciplinary forum to discuss challenges, technologies and emerging directions in system design and implementation that pertain to IoT.

Papers are solicited on a range of topics, including but not limited to:

- Analytic foundations and theory of IoT
- Reliability, security, timeliness, and robustness in IoT systems
- Novel protocols and network abstractions
- Data streaming architectures and data analytics for IoT
- AI/ML for IoT & embedded systems
- IoT-motivated cyber-physical and Industrial IoT (IIoT) systems
- Novel quality requirements and their enforcement mechanisms
- Cloud back-ends and resource management for IoT applications
- Edge and fog computing
- Personal, wearable, and other embedded networked front-ends
- Social computing and human-in-the-loop issues
- Applications for specific domains (smart cities, health, ITS, ...)
- Deployment experiences, case studies & lessons learned
- Evaluation and testbeds
- Energy/power management & harvesting for IoT platforms

General Chairs

Valerie Issarny (INRIA, France)
Haiying Shen (U. Virginia, USA)

Program Chairs

Tao Gu (RMIT, Australia)
Christopher Stewart (Ohio State U., USA)

Steering Committee Chairs

Tarek Abdelzaher (UIUC, USA)
Hui Lei (Futurewei Technologies, USA)

Poster & Demo Chairs

Kang Chen (Southern Illinois U. USA)
Amy Murphy (FBK, Italy)

Publicity Chairs

Jinwei Liu (Florida A&M U., USA)
Georgios Bouloukakis (UC Irvine, USA)
Yuan He (Tsinghua U., China)

Social Media Chair

Lei Yu (IBM Watson Research Center, USA)

Web Chair

Chenxi Qiu (Rowan U., USA)

Publication Chairs

Jia Rao (U. Texas at Arlington, USA)
Li Yan (MIT)

Important Dates

Abstracts due: October 19, 2020

Full papers due: October 26, 2020

Author notification: January 19, 2021



DOI:10.1145/3372114

The value of learning skillsets within a trio of disciplines and the role each plays in DevOps.

BY STUART GALUP, RONALD DATTERO, AND JING QUAN

What Do Agile, Lean, and ITIL Mean to DevOps?

THE METAPHOR OF development teams throwing applications over a wall to the operations group for deployment is often used to vividly illustrate that development and operations operate as silos. The DevOps movement was started in 2008¹⁷ to try and break down these barriers between the development and operations groups. The DevOps movement relies on a culture that strives to understand the capabilities and constraints of the other group (development or operations): “Delivering value to the business through software requires processes and coordination that often span multiple teams across complex systems and involves developing and delivering software with both quality and resiliency.”¹¹

Automation is a major enabler of DevOps as it is highly desirable to automate provisioning, release management, and anything else that is possible.

Continuous build, integration, and delivery are also enablers of DevOps.¹⁷ But DevOps is not just about tools that facilitate development and deployment. In *The DevOps Handbook*,¹⁹ the following myths are debunked.

1. DevOps replaces Agile.
2. DevOps is incompatible with ITIL.
3. DevOps means eliminating IT operations.
4. DevOps is just “infrastructure as code” or automation.

DevOps is about good development practices that continually deliver product features (Agile) effectively with minimal wasted efforts (Lean) which are overseen by good governance controls (Information Technology Service Management, or ITSM).² To that end, a growing consensus within the information technology community is that DevOps = Agile + Lean + ITSM. We believe the integration of Agile, Lean, and ITSM can provide a strong foundation for DevOps.

Adopting DevOps is not an ad hoc or routine operational change, it is transformative in nature and requires a fundamental shift in the traditional ways of working. DevOps is not just a new technology adoption initiative. Rather DevOps adoption should be people-centric, including defining clear roles and providing appropriate training.¹⁴ But the lack of a common

» key insights

- The consensus within the DevOps community is DevOps = Agile + Lean + ITIL.
- The DevOps goal is to enable cross-functional relationships between the development and operations groups thereby enabling them to work together to ensure IT services are transitioned to the live environment successfully.
- The integration of ITIL with Agile and Lean as part of ITIL 4 is a positive step in establishing a practical framework to enable the implementation of DevOps.
- Our data analysis provides significant evidence that there is value gained by IT professionals if they possess Agile (salary premium 26%), Lean (salary premium 9%) and ITIL skills and knowledge (salary premium 16%).



DevOps set of skills and knowledge negatively affects the implementation and training of DevOps roles for organizations and educational institutions.¹⁸ We believe that understanding the value of each discipline (Agile, Lean, and ITSM) will assist organizations and educational institutions to better appreciate the value of the specific skills and knowledge for a DevOps role. Specifically, both IT professionals and organizations can select the best course of action to maximize their investment and time when pursuing and acquiring DevOps skills and talents based on the relative value of each discipline.


In a prior study focusing on the value of Agile skills, Agile skills produced a 22.6% increase in average salary.⁹ In another study focusing on ITIL (the leading ITSM framework⁶), ITIL skills produced an overall salary premium of 10.0% with ITIL certification producing an even greater 14.5% salary premium.¹³

This article begins with an overview of Agile, Lean, and ITSM, emphasizing the areas of overlap, and then addresses the following research questions: (RQ1) Are there compensation benefits for IT professionals that possess Agile, Lean, and ITSM skills? With the logical follow-up question, if there are benefits: (RQ2) What are the estimated benefits?

Agile

After many years of using the waterfall software development methodology (and other less than successful approaches),¹⁷ software developers met to discuss alternative software development methods in 2001. The “Manifesto for Agile Software Development” was the result of this meeting. The Manifesto²³ is a set of four values that are supported by 12 principles. In a 2010 study of Agile practices,⁸ the most widely valued Agile principle was related to business people and developers working together. The 2010 study⁸ found 84% of respondents rated this of high importance. The (tied for) second most valued Agile principle related to achieving customer satisfaction through early and continuous delivery of valuable software. Some 60% of respondents rated this of high importance.

Essentially, the Agile approach is designed to drive shorter feedback



A growing consensus within the information technology community is that DevOps = Agile + Lean + ITSM.

loops that ultimately improve customer value by developing software in a collaborative, iterative, and incremental manner. The main elements of The Manifesto¹⁰ are:

- ▶ Work is done by self-organizing teams, networks, and ecosystems that mobilize the full talents of those doing the work.
- ▶ Work is focused directly on meeting customers’ needs and interaction with the customer is paramount.
- ▶ A “lens” focuses attention on the customers’ needs (when the lens is a person, as in Scrum, the person is known as a “product owner”).
- ▶ Work proceeds in an iterative fashion and progress toward fulfilling the needs of customers is assessed at every stage.

There are a wide range of Agile software development methods including eXtreme Programming, Adaptive Software Development, Scrum, Agile Project Management, Crystal Methods, Feature-Driven Development, Lean Development, and Rational Unified Process.¹ As a result, agility could be considered more of a mindset rather than a specific set of techniques.

Lean

Manufacturing took many concepts from leadership practices and learned how to remove constraints along the flow of work using a variety of short feedback loops.²⁵ “The core idea behind lean is maximizing customer value while minimizing waste,” states The Lean Enterprise Institute.²⁰ The Institute goes on to state that “Simply put, lean means creating more value for customers with fewer resources.”

There has been a long-time connection between Agile and Lean in the IT field. In 2003, Poppendieck and Poppendieck²³ published their book, *Lean Software Development: An Agile Toolkit*. This book was part of Addison-Wesley’s *The Agile Software Development Series*. In their book, Poppendieck and Poppendieck illustrate how 22 different Lean tools, such as seeing waste and value stream mapping, could be applied to the (Agile) software development process.

In 2010, Bell and Orzen⁶ published their book *Lean IT: Enabling and Sustaining Your Lean Transformation*. They applied Lean to the entire IT

organization. In their Lean IT pyramid, the top of their pyramid is culture. The starting building blocks are: consistency of purpose, respect for people, and pursuit of perfection. Intermediate layers include: Voice of the Customer (originally a marketing term widely adopted in business to describe the in-depth process of capturing customer's expectations, preferences, and aversions); quality at the source (a Lean manufacturing principle defines that quality output is not only measured at the end of the production line but at every step of the production process; at each step, the responsibility for quality are the individuals working on the step rather than quality inspectors); systems thinking (a holistic approach to analysis that focuses on the entire system and the relationships between each of the system's constituent parts over time); and flow/pull/Just-in-Time (an approach in which materials, goods, and labor are scheduled to arrive or be replenished only when needed in the process—that is, just in time).

The DevOps Handbook was highly influenced by prior work on Lean and Agile applied to IT. A major theme of this book is the *three ways* described as: “the values and philosophies that frame the processes, procedures, practices of DevOps, as well as the prescriptive steps.”¹⁹

The First Way emphasizes the performance of the entire system, as opposed to the performance of a specific silo of work or department—this can be as large as a division (for example, Development or IT Operations) or as small as an individual contributor (for example, a developer, system administrator).¹⁹

The Second Way is about creating responsive feedback loops. The goal of almost any process improvement initiative is to shorten and amplify feedback loops so necessary corrections can be continually made.¹⁹

The Third Way is about creating a culture that fosters two things: continual experimentation—

taking risks and learning from failure; and understanding that repetition and practice is the prerequisite to mastery.¹⁹

Information Technology Infrastructure Library (ITIL)

Information Technology Service Management (ITSM) is a quality management approach for managing IT services that meet the needs of the business⁷ by focusing on the effective and efficient operation of the IT service provider's internal processes.¹² ITSM is defined as the implementation and management of quality IT services that meet the needs of the business and is performed by IT service providers through an appropriate mix of people, processes, and information technology.¹² There are several ITSM frameworks, but the most widely known framework is the Information Technology Infrastructure Library (ITIL). In our data analysis, we use ITIL knowledge to represent ITSM knowledge.

There are five life cycle stages in ITIL v3: Service Strategy; Service Design; Service Transition; Service Operation; and, Continual Service Improvement (CSI). CSI has many similarities to the Lean concept of kaizen. CSI uses methods from quality management such as the Deming PDCA (Plan-Do-Check-Act) Cycle.¹⁶

Like *The Agile Manifesto*, ITIL explicitly states some essential principles and values. In ITIL v3, each of the five life cycle books begin with a chapter on services and value. In ITIL's definition of a service, it provides an essential core principle—a service is a means of delivering *value* to customers by facilitating *outcomes* customers want to achieve without the ownership of specific costs and risks. ITIL further expounds on this principle as IT service value is composed of two parts: utility and warranty. Utility is a service's fitness for purpose while warranty is a service's fitness for use. Utility is simply a service's functional requirements. Warranty includes availability, capacity, continuity, and security.

In terms of DevOps, Service Design (Development) and Service Operation (Operations) have considerable relevance because of the overlapping activ-

ities in DevOps performed in the processes that make up these two ITIL life cycle stages. Service Design includes processes for Service-Level Management, Availability Management, Capacity Management, IT Service Continuity Management, and Information Security Management. Service Operation consists of five processes: Event Management, Incident Management, Request Fulfillment, Problem Management, and Access Management. ITIL also suggests four generic functions (employee groups): Service Desk, Technical Management, Application Management, and IT Operations Management.¹⁵

The updated ITIL framework, ITIL 4, was released in 2019.³

This name reflects the role ITIL will play in supporting individuals and organizations to navigate the Fourth Industrial Revolution. IT is at the core of every modern business in the global economy. The update will allow ITIL to reflect the fast-paced and complex environment we live in, and new ways of working and emerging practices, all of which are essential not only for ITSM professionals, but also for a wider range of professionals working in the digital transformation world. The purpose of ITIL 4 is to provide organizations with comprehensive guidance for the management of information technology in the modern service economy. ITIL 4 will evolve to provide an end-to-end IT/Digital Operating Model, covering the full delivery (and sustaining) of tech-enabled products and services, guiding how IT interfaces with, and even leads, the wider business strategy.²⁴

ITIL 4 expands on the previous versions by providing a practical and flexible basis to support organizations on their journey to the new world of digital transformation.⁵ It provides an operating model for the delivery and operation of the IT components that fosters team integration. “ITIL 4 also provides a holistic end-to-end picture that integrates frameworks such as Lean, Agile, and DevOps.”²²

Table 1. Median salary for respondents.

Salary: skills and knowledge	n (%)	Median
All respondents	5,081 (100%)	\$85,000
Agile	1933 (38%)	\$100,000
Lean	445 (9%)	\$103,000
ITIL	778 (15%)	\$100,600
Agile and ITIL	375 (7%)	\$111,200
Agile and Lean	290 (6%)	\$112,600
Lean and ITIL	129 (3%)	\$120,000
Agile and Lean and ITIL	99 (2%)	\$124,800
None of the three	2,620 (52%)	\$72,000

Data Collection and Results

A voluntary Web-based survey on salary and skills of IT professionals conducted by Dice (<http://www.dice.com>) was used for this article. The Dice 2018 Tech Salary Report states the survey was administered online by Dice.com, with 10,705 employed technology professionals from the U.S. responding between Oct. 18, 2017 and Dec. 13, 2017. Respondents were invited to participate in the survey in several ways: via an email invitation to Dice’s registered database members, through a notification on the Dice.com home page and/or via site intercept invitations within the site to visitors, and via banner ads on external sites. Additionally, technology professionals who were registered users of eFinancialCareers.com were invited to participate in the survey via an email invitation. A cookie methodology was used to ensure that there was no duplication of responses between or within the various sample groups, and duplicate responses from a single email address were removed. Technology professionals earning salaries of \$350,000 and above were not automatically eliminated from the survey if they met other

criteria (www.DICE.com).

The original dataset contains a certain number of outliers and omissions related to salary. We cleansed the data by setting the minimum wage of \$10.00 per hour or \$20,000 per year (40 hours per week and 50 weeks per year). The records with less than a \$20,000 annual salary were eliminated (most of these eliminated records did not provide a salary figure). We further limited the respondents to only full-time employees in order to obtain a sample of 5,081.

Based on the DICE data, the salaries for IT professionals that possess Agile, Lean, and ITIL skills and knowledge are shown in Table 1. The table provides the salary for all respondents, comparing the salary medians of IT professionals that possess Agile, Lean, or ITIL skills and knowledge to the salary for all respondents.

The results clearly indicate a significant salary premium for possessing Agile, Lean, or ITIL skills and knowledge because the median salaries for individual skills are substantially higher than the overall (all respondents) median. Furthermore, the combination of

Table 2. Human capital model with Agile, Lean, and ITIL (* denotes significant at the .001 level).

	Coefficient
Intercept	11.001*
X	0.026*
X ²	-0.0006*
Agile	0.257*
Lean	0.087*
ITIL	0.158*
F-statistic	192.5*

two skills is even greater and individuals possessing all three skills being the highest. Over half of the respondents did not possess any of the three skills and had substantially lower salaries than their more skilled peers.

Human capital. The dominant economic theory of wage determination is the Human Capital Theory,³ which predicts that differences in wages arise because of differences in human capital which can be accumulated in two main ways—education and experience.¹⁹ Specifically, investments made in one’s occupation are directly correlated to the compensation earned over time that is received for the execution of job tasks. Therefore, educational spending can be considered to be an investment in human capital.²⁷ The more education workers have, the more productive they will be when compared to their less educated counterparts. As a result, the educated worker is more likely to command higher wages. In addition to formal education (indicated by the acquiring of a high school diploma, college degree, and so on), on-the-job training is a very important factor for many IT jobs. Rather than formal education, we factor in whether an IT professional has (self-reported) Agile, Lean, or ITIL skills/knowledge.

As stated earlier, two research questions will be addressed: (RQ1) Are there compensation benefits for IT professionals that possess Agile, Lean, and ITSM skills? With the logical follow-up question, if there are benefits: (RQ2) What are the estimated benefits? Rather than using salary (given in U.S. dollars in the survey) as the dependent variable in a regression model, we employ the natural logarithm of salary because Roy²⁶ showed the natural logarithm of

Regression model.

$$\text{Log } Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \beta_3 \text{Agile}_i + \beta_4 \text{Lean}_i + \beta_5 \text{ITIL}_i + \epsilon_i$$

- where Y_i is the yearly income for each worker,
- β_0 is the intercept term in the regression model that determines the base rate,
- β_1 and β_2 are coefficients that assess the rate of return on experience,
- Agile, is an indicator variable defined to be 1 if the individual possesses Agile skills and 0 if the individual does not possess Agile skills,
- Lean, is an indicator variable defined to be 1 if the individual possesses Lean skills and 0 if the individual does not possess Lean skills,
- ITIL, is an indicator variable defined to be 1 if the individual possesses ITIL skills and 0 if the individual does not possess ITIL skills,
- $\beta_3, \beta_4,$ and β_5 are coefficients that assess the rate of return on Agile, Lean, and ITIL knowledge, respectively, and ϵ_i is the random disturbance associated with the i -th worker.

salary rather than salary produced a much better model fit. As stated earlier, experience is often incorporated in human capital models so it will also be one of our independent variables. Mincer²¹ showed that experience should be modeled as concave because as experience reaches a certain point, salary cannot increase indefinitely. Therefore, experience squared will be another independent variable. In addition, three dummy or indicator independent variables will be used to model whether each respondent indicated whether they possess Agile, Lean, or ITIL skills. This set of variables produce the the regression model depicted in the accompanying figure. We fitted the regression equation using the 5,081 respondents from the cleansed DICE sample with the results given in Table 2.

All coefficients and the overall model are highly significant. Therefore, the answers to the research questions are clearly affirmative based on the Human Capital Model. The fact that the coefficients of Agile, Lean, and ITSM skills in the estimated regression equation are positive provides the answer to RQ1 about compensation benefits for IT professionals that possess these skills. The salary premium is computed by taking each coefficient’s inverse of the natural logarithm function—that is, each coefficient is plugged into the exponential function. Agile skills, by far, produce the greatest salary premium (26%) with ITIL (16%) and Lean (9%) skills and knowledge producing significant overall salary premiums. Given the logarithmic nature of the regression model, one can multiply these numbers to get the premiums for pairs of skills and all three skills. The largest salary premium would be produced by the combination of all three skills. This answers RQ2 about the estimated benefits of these skills.

Conclusion

The goal of DevOps is to enable cross-functional relationships between the development and operations groups thereby enabling the two groups to work together to ensure IT services are transitioned to the live environment without problems. The specific skills and knowledge needed for a DevOps implementation will vary based on the infrastructure and business focus. This

adds to the challenge of implementing DevOps because there is a clear lack of universally accepted skills and knowledge requirements. The growing consensus within the DevOps community is that DevOps = Agile + Lean + ITIL helps to establish a common set of base skills and knowledge that transcend business environments and toolchains. As reflected by the salary premiums, our data analysis provides significant evidence that there is value gained by IT professionals if they possess Agile (salary premium 26%), Lean (salary premium 9%), and ITIL skills and knowledge (salary premium 16%). Organizations and educational institutions that focus on cultivating these skills and knowledge will enhance the IT professional’s ability to build cross-functional processes and also use appropriate technology to enhance an overall collaborative automated DevOps environment.¹⁴

ITIL’s Service Design and Service Operation processes can clearly be adapted for DevOps as these processes and the generic functions will still be necessary. Similarly, Agile can adapt ITIL Service Transition processes that help monitor and control service delivery, such as Change Management, Service Asset and Configuration Management, and Release and Deployment Management. In addition, Lean workflow concepts can better improve most (if not all) ITIL processes.

The integration of ITIL with Agile and Lean as part of ITIL 4 is a positive step in the direction of establishing a practical framework to enable the implementation of DevOps. Kim better summarizes our opinion as follows: “For many years, I’ve felt I’ve been the official ITIL® apologist in the DevOps community, because I’ve always believed that DevOps and ITIL should be able to peacefully coexist. But these days, I feel that a more activist role in the DevOps community is necessary—we must reach out and form effective bridges with the ITIL community, because ITIL is the most powerful and entrenched orthodoxy in large, complex IT organizations.”¹⁹

References

1. Abrahamsson, P., Oza, N. and Siponen, M.T. Agile software development methods: A comparative review. *Agile Software Development* (2010), 31–59.
2. Anand, A. AXELOS Lean Enterprise Podcast (2017).
3. AXELOS.com; <https://www.axelos.com>
4. Beach, C. and Berndt, E.R. The practice of econometrics:

- Classic and contemporary. *Can. J. Econ.* 26, 3 (1993), 750.
5. Beadle, A. Getting Ready for ITIL 4, The Next Evolution of ITIL. Learning Tree International, 2018; www.learningtree.com
6. Bell, S.C. and Orzen, M.A. *Lean IT: Enabling and Sustaining Your Lean Transformation*. CRC Press, 2016.
7. van Bon, J., Kemmerling, G. and Pondman, D. *IT Service Management: An Introduction*, 2002.
8. de Cesare, S., Lycett, M., Macredie, R.D., Patel, C. and Paul, R. Examining perceptions of agility in software development practice. *Commun. ACM* 53, 6 (June 2010), 126.
9. Dattero, R., Galup, S.D., Kan, A. and Quan, J. It pays to be agile. *J. Computer Information Systems* 57, 3 (2016), 252–257.
10. Denning, S. Agile: It’s time to put it to use to manage business complexity. *Strategy & Leadership* 43, 5 (2015), 10–17.
11. Forsgren, N. and Kersten, M. DevOps metrics. *Commun. ACM* 61, 4 (Apr. 2018), 44–48.
12. Galup, S.D., Dattero, R., Quan, J.J. and Conger, S. An overview of IT service management. *Commun. ACM* 52, 5 (May 2009), 124.
13. Galup, S.D., Dattero, R. and Quan, J.J. The compensation benefit of ITIL® skills and certifications. *Intern. J. Service Science, Management, Engineering, and Technology* 7, 2 (2016), 1–15.
14. Gill, A.Q., Loumish, A., Riyat, I. and Han, S. DevOps for information management systems. *VINE J. Info. Knowledge Management Systems* 48 (2018), 122–139; <https://doi.org/10.1108/vjikms-02-2017-0007>
15. Great Britain. Cabinet Office and Great Britain. Stationery Office. ITIL Service Operation. Stationery Office/Tso, 2011.
16. Great Britain. Cabinet Office and Stationery Office. ITIL Continual Service Improvement. Stationery Office/Tso, 2011.
17. Haddad, C. *DevOps = DevOps Principles + DevOps Practices - DZone DevOps*, 2014. dzone.com/articles/devops-devops-principles
18. Kerzazi, N. and Adams, B. 2016. Who needs release and devops engineers, and why? In *Proceedings of the Intern. Workshop on Continuous Software Evolution and Delivery*, 2016. DOI:<https://doi.org/10.1145/2896941.2896957>
19. Kim, G., Debois, P., Willis, J. and Humble, J. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution 2016.
20. Lean Enterprise Institute | Lean Production | Lean Manufacturing | LEI | Lean Services; <https://www.lean.org>
21. Mincer, J. *Schooling, Experience, and Earnings*. 1974.
22. Pink Elephant; <https://www.pinkelephant.com/en-us/>
23. Poppendieck, M. and Poppendieck, T. *Lean Software Development: An Agile Toolkit: An Agile Toolkit*. Addison-Wesley, Reading, PA, 2003.
24. Quint Wellington Redwood. ITIL 4 FAQs | Quint Group; <https://www.quintgroup.com/en-us/insights/itil4-frequently-asked-questions/>
25. Ranger. Ranger4 | DevOps Evolution Experts; <https://www.ranger4.com>
26. Roy, A.D. The distribution of earnings and of individual output. *Econ. J. Nepal* 60, 239 (1950), 489.
27. Schultz, T.W. Capital formation by education. *J. Polit. Econ.* 68, 6 (1960), 571–583.

Stuart Galup (sgalup@fau.edu) is an associate professor in the Information Technology and Operations Management Department at Florida Atlantic University, Boca Raton, FL, USA.

Ronald Dattero (RonDattero@Missouristate.edu) is professor emeritus at Missouri State University, Springfield, MO, USA.

Jing Quan (jxquan@salisbury.edu) is a professor and chair of the Department of Information and Decision Sciences at Salisbury University, Salisbury, MD, USA.

© 2020 ACM 0001-0782/20/10 \$15.00



Watch the authors discuss this work in an exclusive *Communications* video. <https://caom.acm.org/videos/agile-lean-and-itil>

DOI:10.1145/3375545

The story of the development of a sound, static method for worst-case execution-time analysis.

BY REINHARD WILHELM

Real Time Spent on Real Time

THE GENERAL SETTING for worst-case execution time (WCET) analysis is that a set of hard real-time tasks is to be executed on a given hardware platform. Hard real-time tasks have associated deadlines within which they must finish their execution. The deadlines may be given by periods. *Timing verification* must verify these timing constraints are satisfied. Traditionally, timing verification is split into a *WCET analysis*, which determines upper bounds on the execution times of all tasks, and a *schedulability analysis*, which takes these upper bounds and attempts to verify the given set of tasks when executed on the given platform will all respect their deadlines.

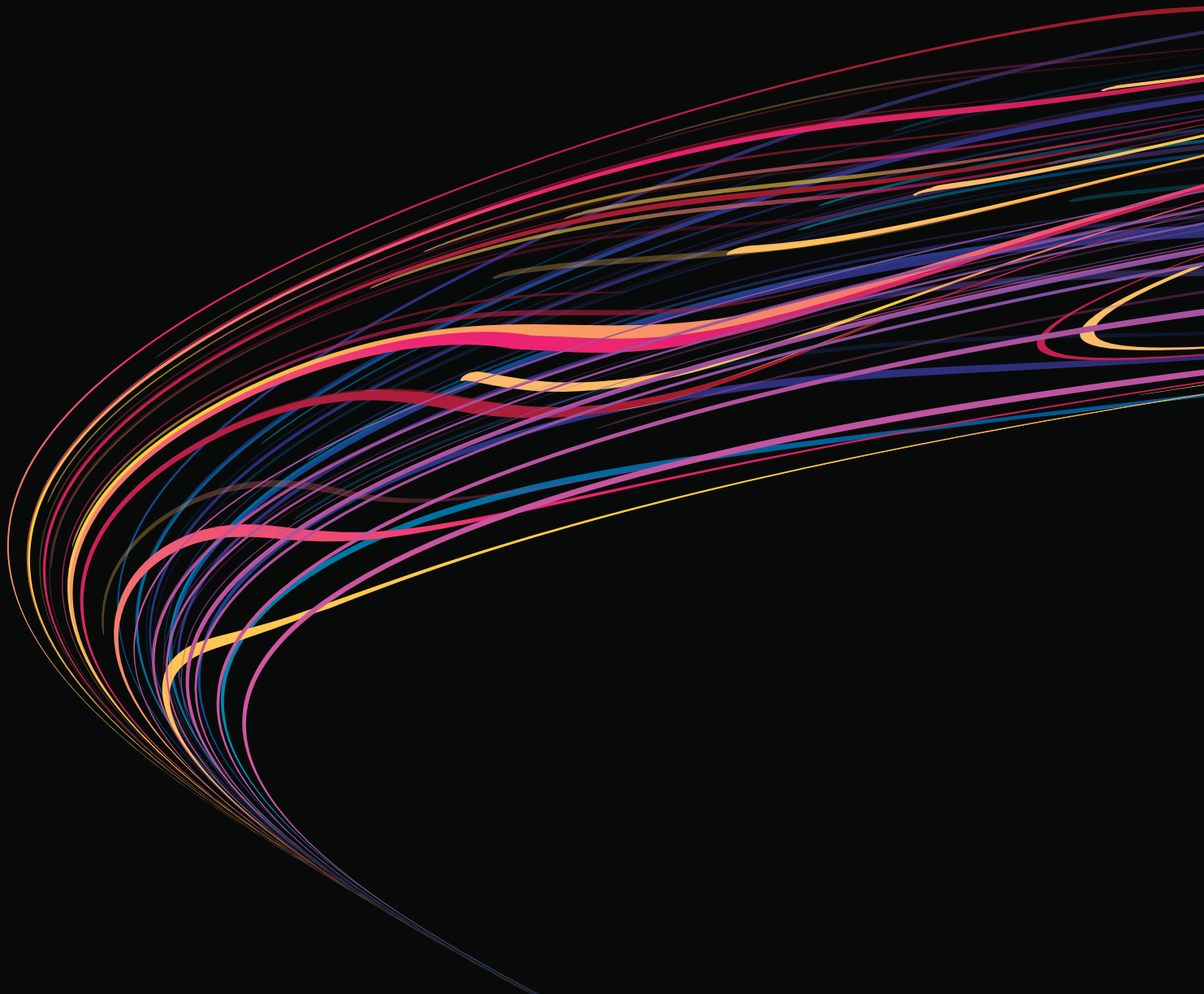
The problem to determine upper (and potentially also) lower bounds on execution times underwent a transition in the 1990s:

► In the old days, textbooks about the realization of real-time systems would strongly argue against the use of execution platforms with caches, pipelines, and such. For previously used architectures with instructions that had constant execution times, WCET analysis methods using *timing schemata*²³ were the method of choice. Timing schemata describe how (bounds on) the execution times of a programming-language construct were composed from the (bounds on) the execution times of its components. These methods would thus do structural induction over the structure of a program and determine bounds for ever bigger parts of the program. Worse yet, industry's "best practice" was, and unfortunately partly still is, to do some end-to-end measurements, ignore some unwelcome outliers, if optimism prevailed, or add some safety margin, if more problem-awareness dominated.

► The introduction of performance-enhancing architectural components and features such as caches, pipelines, and speculation made methods based on timing schemata obsolete. Execution times did not compose any longer because instruction execution times were now dependent on the execution state in which they were executed. In the composition $A;B$, the execution time of statement B depended on the execution state produced by statement A . The variability of execution times

» key insights

- WCET searches a huge state space for a longest path. Adequate abstraction of the execution platform is key to cope with the complexity of the analysis, and Abstract Interpretation provides the theoretical foundation for a sound and efficient WCET analysis.
- The Timing Predictability of an architecture determines the efficiency of WCET analysis and the precision of its results.
- Some performance-enhancing features ruin timing predictability and at the same time open the door to hardware security-attacks like Spectre and Meltdown.



grew with several architectural parameters, for example, the cache-miss penalty and the costs for pipeline stalls and for control-flow mispredictions.

The introduction of multicore execution platforms into the embedded real-time domain made the problem still more difficult. These platforms typically have shared resources, and the interference on these shared resources complicates the determination of upper execution-time bounds for tasks executed on such a platform. A few words about terminology. From the beginning we aimed at sound WCET-analysis methods. The results of a sound WCET analysis are *conservative*, that is, they will never be exceeded by an execution. We consider being

conservative as a Boolean property. Often conservative is used as a metric, being more conservative meaning being less accurate. For an unsound method, however, it does not make sense to speak about being more or less conservative. It is not even clear whether being “more conservative” means moving toward the real WCET from below or moving further away from the real WCET by increasing overestimation. The second, quite important property, referred to when mentioning conservatism, is *accuracy* of the results of a WCET analysis.

WCET analysis can be seen as the search for a longest path in the state space spanned by the program under analysis and by the architectural plat-

form. The analysis is based on the assumption that the analyzed programs terminate, that is, all recursions and iterations are bounded. We are not confronted with the undecidability of the halting problem. In trying to determine bounds on recursion or iteration in a program, our tool might discover that it cannot determine all of the bounds and will ask the user for annotations. All WCET bounds are then valid with respect to the given annotations.

This state space is thus finite, but too large to be exhaustively explored. Therefore, (safe) overapproximation is used in several places. In particular, an abstraction of the execution platform is employed by the WCET analysis. We will in the following cover static

analyses of the behavior of several architectural components.

Cache Analysis

Our engagement in timing analysis started with the dissertation work of Christian Ferdinand at around 1995. I had proposed several thesis topics, which he all rejected with the argument, “This is of no interest to anybody,” meaning irrelevant for industrial practice. When I proposed to develop an analysis of the cache behavior he answered, “This may actually be of interest to somebody.” He was able to convince himself (and me) very quickly that an idea we had would work. He used the program-analysis generator PAG, conceived by Martin Alt and realized by Florian Martin in his Ph.D. thesis,¹⁹ to implement a prototype cache analysis for caches with a *least-recently used* (LRU) replacement strategy. This was, and still is, WCET researcher’s dearest replacement policy. Our first submitted article on cache analysis¹ confirmed Ferdinand’s appreciation for the subject. It received enthusiastic reviews as for the relevance of the problem we had solved and for the elegance of the solution.

Unlike existing methods, Ferdinand designed two different abstract domains for cache analysis, a *must* and a *may* domain.^{1,6,8} An abstract *must* cache at a program point indicates which memory blocks will be in all concrete caches whenever execution reaches this program point. There is an underlying assumption that program execution is not disturbed by interrupts or preemption. Otherwise, the impact of interrupts or preemptions must be taken into account by analyzing the maximal cache impact through a CRPD analysis.²

An abstract *must* cache state computed at a program point represents an over-approximation of the set of concrete cache states that may reach this program point. All the concrete cache states in this overapproximation have as common contents the memory blocks that are sure to be in the concrete cache when execution reaches this program point. In LRU caches, memory blocks logically have an *age*. The age is between 0 and the *associativity*—1 of the cache (set) and corresponds to the relative position in the

access sequence of cached memory blocks, extended by some way to indicate *not cached*. The associativity of a fully associative cache is equal to its capacity. For a set associative cache it is the size of a cache set, that is, the number of memory blocks fitting into each set. It was (for us) easy to see our analysis should perform a kind of intersection and associate with the elements in the resulting set their maximal age from the incoming *must* caches, whenever control flow merges. Abstract *must* caches can be used to predict cache hits.

An abstract *may* cache at a program point indicates which memory blocks may be in a concrete cache whenever execution reaches this program point. In analogy, our analysis uses union at control-flow merge points and associates the minimal incoming age with the elements in the union. Taking the complement of a *may* cache gives the information which memory blocks will be in no concrete cache arriving at this program point. It thus allows to predict cache misses. In Lv,¹⁸ we called such analyses *classifying analyses* as their results allow to classify some memory accesses as either definite hits or definite misses.

In contrast, *persistence analyses* are called *bounding analyses*. They aim at bounding the number of cache reloads of memory blocks. Intuitively, a memory block is called persistent if it suffers at most one cache miss during program execution. We defined such a persistence analysis, which was too beautiful to be correct. The tempting idea was to add another cache line with age *associativity* in which all memory blocks were collected that had been replaced in the cache at least once and to compute the maximal position (relative age) for all memory blocks that may be in the cache. The analysis would thus use union and maximal age at control-flow merge points. Our mistake was that we ignored the capacity constraints of the caches. Our analysis could collect more memory blocks in an abstract cache than would fit into the concrete cache and thereby underestimate the age. Luckily, this cache-persistence analysis was never implemented in AbsInt’s tools. The error was later corrected by Cullmann and Huynh.^{4,15}

Let me jump out of the story to some later developments: Jan Reineke has clarified the semantic foundations of persistence analysis.²¹ All types of persistence are identified by the occurrence of certain patterns in memory-access traces, while categorizing cache analyses, that is, *must* and *may* analyses only abstract from the last state in a trace. Reineke also identified a whole zoo of different persistence analyses. End of excursion.

Understanding Our Approach

We developed the cache analysis with the goal of classifying memory accesses as either definite cache hits or definite cache misses. The difference to competing approaches was that we could describe our cache analyses as abstract interpretations.³ This meant we defined the following:

- ▶ *domains of abstract cache states* with a *partial order* representing which domain elements contained better information than other elements,
- ▶ corresponding to this partial order, a *join function*, used to combine incoming abstract domain elements at control-flow merge points, for example, some kind of intersection for the *must-cache analysis*, and,
- ▶ *abstract cache effects* for each memory access, describing the update of abstract cache states corresponding to this memory access.

This stood in stark contrast to the state of the art in cache analysis, which would typically give a page of pseudo-C code and claim that this code would implement a sound cache analysis, of course without any correctness arguments!

Now, that we had solved one sub-problem of WCET analysis, it was time to reflect more deeply what the essence of our method was, and we identified the following central idea behind our WCET-analysis method:

- ▶ Consider any architectural effect that lets an instruction execute longer than its fastest execution time as a *timing accident*. Typically such timing accidents are cache misses, pipeline stalls, bus-access conflicts, and branch mis-predictions. Each such timing accident had to be paid for, in terms of execution-time cycles, by an associated timing penalty. The size of a timing penalty can be constant but

may also depend on the execution state. We consider the property that a particular instruction will not cause a particular timing accident as a safety property. The occurrence of a timing accident thus violates a corresponding safety property.

► Use an appropriate method for the verification of safety properties to prove that for individual instructions in the program some of the potential timing accidents will never happen. Reduce the worst-case execution-time bound for an instruction, which a sound WCET analysis would have to assume, by the penalties for the excluded timing accidents.


► Abstract interpretation³ is a powerful method to prove safety properties. Use it to compute certain invariants at each program point, namely an upper approximation of the set of execution states that are possible when execution reaches this program point. Derive safety properties, that certain timing accidents will not happen, from these invariants.

This method for the micro-architectural analysis was the central innovation that made our WCET analysis work and scale.


Our First Illusions

Christian Ferdinand had finished his very fine dissertation on cache analysis in 1997. It still represents the state of the art in cache analysis for LRU caches.⁶ Since everybody else working in the area had tried to solve this problem first, and we were convinced that our solution was the best, we felt that we had essentially solved the WCET-analysis problem. Very optimistically we founded AbsInt^a early in 1998, “we,” being five former or actual Ph.D. students and me.

This optimism turned out as wrong in several aspects. Firstly, more or less nobody uses LRU caches in their processors since the logic is considered too complex. Frequently used replacement policies are PLRU, FIFO, random replacement, or even strange looking approximations of random replacement like in the Motorola Coldfire, which is flying in the Airbus A340, and which Airbus selected as a real-life processor to test our ap-



All types of persistence are identified by the occurrence of certain patterns in memory access traces, while categorizing cache analyses, that is, must or may analyses only abstract from the last state in a trace.



proach.⁷ This processor had a four-way set-associative cache with a rather strange cache-replacement strategy using one global round-robin replacement counter keeping track of where the next replacement should take place. It meant our cache analysis could essentially only keep track of the last loaded cache line in each cache set. This strange beast of a cache triggered the concept of *timing predictability*, which turned out to be a very fruitful research area, to be discussed later.

There was a second assumption that turned out to be false. It is intuitively clear the accuracy of the cache analysis has a strong impact on the accuracy of the WCET-analysis results. This led us to believe that cache analysis was the difficult part of WCET analysis, that the rest would be easy. It turned out that caches were relatively easy to analyze, although good solutions for Non-LRU caches had yet to be found. Much later, between 2008–2011, Daniel Grund developed an efficient analysis for FIFO caches and approached a solution for the analysis problem for PLRU caches.^{9–11}

Pipelines were much more difficult to analyze and also appeared in more variations and were mostly badly documented. And then Airbus informed us about the existence of peripheries and system controllers, architectural components that serious WCET researchers had never heard of. Their analyses can have a very strong influence on the accuracy of timing analyses.

Control-Flow Reconstruction

But first we needed to get programs to analyze. WCET analysis has to be done on the executable level because the compiler influences the execution time by its memory allocation and code generation. So, we needed a reconstruction of the control flow from binary programs. This was part of Henrik Theiling’s Ph.D. thesis.²⁴ Decoding individual instructions is relatively easy, but identifying the control flow, in particular switch tables is a non-trivial task.

Pipeline Analysis

At the end of the 1990s, Stephan Thesing started to develop the framework for modeling pipeline architectures.^{16,25}


^a www.absint.com

He also did, as far as I know, the first modeling of a system controller for WCET analysis.²⁶ Doing this precisely was highly important because an imprecise, too abstract model of a systems controller can easily cost an order of magnitude more accuracy than an imprecise pipeline model.


Pipeline analysis is a highly complex part of the overall analysis because, unlike caches, most pipelines do not have compact, efficiently updatable abstract domains. Cache analysis is efficient because the sets of concrete cache states that may occur at a program point can be compactly represented by one abstract cache state, and abstract cache states can be efficiently updated when a memory access is analyzed. Essentially, pipeline analysis uses an expensive powerset domain, that is, it collects sets of pipeline states instead of computing an abstract pipeline state representing sets of pipeline states. This characteristic would, in principle, make it amenable to model checking. Stephan Wilhelm tried this around 2009 and encountered severe problems in the application of model checking to pipeline analysis.^{28,29} In particular, interfacing symbolic representations of pipelines with abstract representations of caches, while preserving accuracy, is difficult. Daniel Kaestner's Astrée group made a similar experience when attempting to interface Astrée with some model checkers.^b It appeared that model checking and abstract interpretation were communicating badly and thus seemed to replicate the behavior of their inventors.

Another excursion into the future: Hahn et al.¹³ describes a strictly in-order pipeline providing for compact abstract domains. Strictly in-order pipelines avoid all downstream dependences between consecutive instructions, such as the one of an operand load of an earlier instruction on the instruction fetch of a consecutive instruction. In case of a contention, the operand load is always guaranteed to be executed first. The loss in (average-case) performance compared to a traditional in-order pipeline was measured to be between 6%

^b <https://www.absint.com/astree/index.htm>



Pipeline analysis is a highly complex part of the overall analysis because, unlike caches, most pipelines do not have compact, efficiently updatable abstract domains.



and 7%,¹² but the analysis efficiency increased by an order of magnitude.

The Breakthrough

The European project *Daedalus, Validation of critical software by static analysis and abstract testing*, which ran from 2000 to 2002, associated us with an extremely valuable partner, Airbus. Patrick Cousot had organized an industry seminar on abstract interpretation. One of the participants was an Airbus engineer, Famantanantsoa Randimbivololona, in charge of identifying new, usable and helpful methods and tools for software development at Airbus. Randim had listed the most severe of Airbus' problems, and Cousot composed a consortium for a European project targeted at solving these problems. Saarbrücken was listed with *WCET Analysis*. My admiration of the problem-awareness of the software developers at Airbus grew during my first visit to the Airbus headquarters and development labs in Toulouse, France. Everybody greeted me, expressing their concern that they had no viable solution for the verification of their real-time requirements and their hope that we would provide a solution.

This is a good point to describe previous funding. In *DFG Collaborative Research Center 124*, DFG (our National Science Foundation) had funded the development of the foundations of WCET analysis, including Florian Martin's Program Analyzer Generator (PAG). When this DFG Collaborative Research Center, which has run 15 years (from 1983 to 1998) approached its end, DFG had just initiated a new type of grant, a *Transfer Center*, meant to support transfer of results of successful Research Centers to practice. This was a perfect fit for our situation. We applied and were granted solid funding for further development. At about this time, Airbus was searching for a solution for their WCET-analysis problem. Cousot formed the consortium, and the EU Commission granted us the Daedalus project. In hindsight, this sequence of funded projects appears like a miracle, each at exactly the right time!

Back to our contacts with Airbus and their search for some one to solve the WCET problem: They knew their previously used measurement-based

method, also used in certification, did not work any longer for the execution platform selected for the Airbus A380, namely the Motorola MPC755.

The Airbus people provided us with benchmark software, a set of 12 denatured tasks, each consisting of several million instructions, as they were flying them in the A340. The platform there was the Motorola Coldfire processor, mentioned earlier. The tool we developed until 2001 was able to analyze the benchmark provided by Airbus in decent time and with quite precise results. The upper bounds our tool computed made the Airbus people quite happy because they were approximately in the middle between the worst observed execution times and the upper bound determined by Airbus with a measurement-based method using safety margins. More precisely, our analysis results were overestimating the worst observed execution times by roughly 15%. This breakthrough was reported in Ferdinand et al.⁷ Something surprising, at least for me, happened when I described our approach and reported our results at EMSOFT 2001. Highly appreciated colleagues like Hermann Kopetz and Gérard Berry were storming the stage and congratulated me as if I had just won an Oscar. Indeed, this paper received the EMSOFT Test-of-Time Award 2019.

Some words about our long-time cooperation partner, Airbus, in Toulouse. We have experienced the group in charge of developing safety-critical software as problem-aware, highly competent, and extremely cooperative. They wanted to have this problem solved and they trusted us to solve it, and they kept us focused on the real problems, and thus prevented us from solving simplified, self-posed problems, a tendency academic researchers often are inclined to follow.

As a result of our successful development, Airbus offered our tools to the certification authorities for the certification of several Airbus plane generations, starting with the Airbus A380. The European Union Aviation Safety Agency (EASA) has accepted the AbsInt WCET analysis tool as validated tool for several time-critical subsystems of these plane types. We were less successful with Airbus' competitor, who partly certifies their planes themselves,

as it recently turned out, and with the certification authority in charge, who doesn't seem to require the use of a sound verification technology for real-time requirements.

Predictability

When modeling the Motorola Coldfire cache we noticed that only one-fourth of its cache capacity could be predicted. This quickly led us to consider the problem of *timing-predictability* of architectures.¹⁴ In his Ph.D. thesis, Jan Reineke developed the first formally founded notion of predictability, namely that of cache predictability.^{20,22} The concept behind this notion is that a cache architecture, more precisely its cache-replacement policy is more predictable than another one if it recovers from uncertainty about cache contents faster, that is, needs fewer memory accesses to remove uncertainty from the abstract cache. Among all the considered cache-replacement strategies LRU fares provably best.

Reineke also compared how *sensitive* caches are to changes to the initial cache state. He could show that all non-LRU cache replacement strategies he considered were quite sensitive to such changes. This means the difference in the cache-miss rate is only bounded by the length of the memory-access sequence. Thus, missing an initial cache state when measuring execution time may mean to miss a memory-access sequence with a high cache-miss rate.

In Wilhelm et al.,²⁷ we collected our wisdom concerning timing predictability of several types of architectural components. It heavily influenced the design of the Kalray MPPA.⁵

One could at this point remark that a future automatically driven car will employ a GPU executing learned-pattern recognition, which is controlled by an 8-core-ARM architecture whose design contradicts under almost all aspects this collected wisdom of ours. It employs random-replacement caches, a cache coherence protocol, a shared bus, and all DRAM memory.

Another remark is in place here. Our efforts to push the predictability issue had limited effect. In retrospect, it looks like we came up too early with our complaints and the ideas to remedy the corresponding problems. A few

years later hardware security-attacks like Meltdown and Spectre showed that architectural features with low predictability were also the basis for these security attacks. A combination of timing unpredictability and vulnerability to security attacks might have discredited the architectural components more effectively.

AbsInt

WCET analysis for single-core architectures had been essentially solved by a sequence of Ph.D. theses in my group. The only import was the Implicit Path Enumeration Technique (IPET) of Li and Malik.¹⁷ Li and Malik had managed to model the timing behavior of programs and the entire architecture as an integer linear program, which was far from efficiently solvable. The IPET technique was adopted to our setting by Henrik Theiling in his Ph.D. thesis.²⁴

It may be valuable information for many readers to estimate the necessary effort to develop a sound formal method for an industrially relevant non-trivial problem more or less from scratch. At the core of our work leading to the first usable tool, as described in Ferdinand et al.,⁷ were three Ph.D. theses, those of Christian Ferdinand, Stephan Thesing, and Henrik Theiling. Their joint development effort would amount to approximately 11 person years, ignoring the effort required to write up the theses, publish results, and satisfy project requirements. Another three years went into the implementation of static cache analysis for some non-LRU caches and into the value analysis of the timing-analysis tool. The latter was based on existing theory developed in Cousot and Cousot³ but had to be adapted to the analysis of binary executables and to all the peculiarities of a machine semantics. Altogether the effort invested in the first usable tool would add up to roughly 14 person years. However, one should not underestimate the accelerating effect that PAG¹⁹ had on the implementation of and experimentation with several abstract interpretations within the timing-analysis tool. This development effort was followed by work to improve efficiency of the analyses and accuracy of the results, by research into the predictability of architectural features and, based on the


results, into ways to exploit the configurability of processor architectures.

We had founded AbsInt to industrialize our WCET technology. Well, we solved the problem, we had instantiations for some processor architectures, basically for those that Airbus and their suppliers needed. However, we had to learn that hardly any two potential customers employed the same architecture configuration. The decision for a new platform was taken without considering whether a WCET-analysis existed for this platform. Instantiating our technology for a new, complex platform took a lot of effort, and platforms were not getting simpler! In consequence, such an instantiation was very expensive, which did not raise the motivation of potential customers to buy our WCET tools or order the development of a new instance for their platform. In addition, there existed some competitors, who marketed their measurement-based, unsound timing analysis and often forgot to mention the unsoundness of their tool. When companies that developed or integrated hard real-time systems were obliged to show “that they did something about this nasty problem” this unsound, inexpensive solution was sometimes preferred to show “that we do something” (and didn’t pay too much for it). So, industrializing and marketing a sound WCET technology, that inherently needed to be expensive, was no promising way to get rich.

However, our development of a sound method that actually solved a real problem of real industry was considered a major success story for the often disputed formal-methods domain. AbsInt became the favorite partner for the industrialization of academic prototypes. First, Patrick Cousot and his team offered their prototype of Astrée, a static analysis for run-time errors, which in cooperation with some of the developers has been largely extended by AbsInt. Then, Xavier Leroy offered the result of his much-acclaimed research project, CompCert, the first verified optimizing C compiler. Both Astrée and CompCert are now AbsInt products.

Conclusion

My former Ph.D. students and I have solved a relevant, non-trivial problem,

namely how to determine reliable and precise upper bounds on execution times of programs. We were not the only ones to attempt this. Why were we more successful than other groups? Essentially the answer is, we had a firm background in formal methods, particularly in abstract interpretation, and abstraction of the execution platform played the decisive role in our approach. Without the right abstraction of the architecture, the search space is just too large. However, there is more to it! WCET analysis consists of many phases. A practically usable WCET-analysis method requires strong solutions to all the subproblems and their adequate interaction. Otherwise, either the effort is too high, or the accuracy is too low. The people at AbsInt did an excellent engineering job to come up with WCET-analysis tools and later also other tools that were usable on an industrial scale. 

References

- Alt, M., Ferdinand, C., Martin, F., and Wilhelm, R. Cache behavior prediction by abstract interpretation. R. Cousot and D.A. Schmidt, Eds. In *Proceedings of Static Analysis, 3rd Intern. Symp.* (Aachen, Germany, Sept. 24–26, 1996). LNCS 1145, 52–66; https://doi.org/10.1007/3-540-61739-6_33
- Altmeyer, S., Maiza, C., and Reineke, J. Resilience analysis: Tightening the CRPD bound for set-associative caches. J. Lee and B.R. Childers, Eds. In *Proceedings of the ACM SIGPLAN/SIGBED 2010 Conf. Languages, Compilers, and Tools for Embedded Systems*, (Stockholm, Sweden, Apr. 13–15, 2010), 153–162; doi:10.1145/1755888.1755911.
- Cousot, P. and Cousot, R. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. R.M. Graham, M.A. Harrison, and R. Sethi, Eds. In *Proceedings of the 4th ACM Symp. on Principles of Programming Languages*, (Los Angeles, CA, USA, Jan. 1977), A8–252; doi:10.1145/512950.512973.
- Cullmann, C. Cache persistence analysis: Theory and practice. *ACM Trans. Embedded Comput. Syst.* 12, 1 (2013), 40:1–40:25; doi:10.1145/2435227.2435236.
- de Dinechin, B.D., van Amstel, D., Poulhiès, M., and Lager, G. Time-critical computing on a single-chip massively parallel processor. G.P. Fettweis and W.N., Eds. *Design, Automation & Test in Europe Conf. & Exhibition* (Dresden, Germany, Mar. 24–28, 2014), 1–6; doi: 10.7873/DATE.2014.110.
- Ferdinand, C. *Cache Behavior Prediction for Real-Time Systems*. Pirrot, 1997; <http://d-nb.info/953983706>.
- Ferdinand, C., Heckmann, R., Langenbach, M., Martin, F., Schmidt, M., Theiling, H., Thesing, S., and Wilhelm, R. Reliable and precise WCET determination for a real-life processor. *EMSOFT LNCS*, 2211 (2001), 469–485.
- Ferdinand, C. and Wilhelm, R. Efficient and precise cache behavior prediction for real-time systems. *Real-Time Systems* 17, 2–3 (1999), 131–181.
- Grund, D. *Static Cache Analysis for Real-Time Systems: LRU, FIFO, PLRU*. Ph.D. thesis, Saarland University, 2011.
- Grund, D. and Reineke, J. Abstract interpretation of FIFO replacement. J. Palsberg and Z. Su, Eds. In *Proceedings of the 16th Intern. Symp. Statist. Analysis* (Los Angeles, CA, USA, Aug. 9–11, 2009). LNCS 5673; doi:10.1007/978-3-642-03237-0_10.
- Grund, D. and Reineke, J. Toward precise PLRU cache analysis. B. Lisper, Ed. In *Proceedings of the 10th Intern. Workshop on Worst-Case Execution Time Analysis*, (Brussels, Belgium, July 8, 2010). *OASICS 15* (2010) 23–35. Schloss Dagstuhl - Leibniz-

Zentrum fuer Informatik, Germany; doi:10.4230/OASICS.WCET.2010.23.

- Hahn, S. On Static Execution-Time Analysis - Compositionality, Pipeline Abstraction, and Predictable Hardware. Ph.D. thesis, Saarland University, 2019.
- Hahn, S., Reineke, J., and Wilhelm, R. Toward compact abstractions for processor pipelines. R. Meyer, A. Platzer, and H. Wehrheim, Eds. In *Proceedings of Correct System Design—Symp. Honor of Ernst-Rüdiger Olderog* (Oldenburg, Germany, Sept. 8–9, 2015). LNCS 9360, 205–220; doi:10.1007/978-3-319-23506-6_14.
- Heckmann, R., Langenbach, M., Thesing, S., and Wilhelm, R. The influence of processor architecture on the design and the results of WCET tools. In *IEEE Proceedings on Real-Time Systems* 91, 7 (2003), 1038–1054.
- Huynh, B.K., Ju, L., and Roychoudhury, A. Scope-aware data cache analysis for WCET estimation. In *Proceedings of the 17th IEEE Real-Time and Embedded Technology and Applications Symp.* (Chicago, IL, USA, Apr. 11–14, 2011) 203–212; doi:10.1109/RTAS.2011.27.
- Langenbach, M., Thesing, S., and Heckmann, R. Pipeline modeling for timing analysis. M.V. Hermenegildo and G. Puebla, Eds. In *Proceedings of the 9th Intern. Symp. on Static Analysis* (Madrid, Spain, Sept. 17–20, 2002), LNCS 2477, 294–309; doi:10.1007/3-540-45789-5_22.
- Li, Y.S. and Malik, S. Performance analysis of embedded software using implicit path enumeration. In *Proceedings of the 32nd ACM/IEEE Design Automation Conf.* (June 1995), 456–461.
- Lv, M., Guan, N., Reineke, J., Wilhelm, R., and Yi, W. A survey on static cache analysis for real-time systems. *LITES* 3, 1 (2016), 05:1–05:48; doi:10.4230/LITES-v003-i001-a005.
- Martin, F. *Generating Program Analyzers*. Ph.D. thesis, Saarland University, Saarbrücken, Germany, 1999; <http://scidok.sulb.uni-saarland.de/volltexte/2004/203/index.html>.
- Reineke, J. *Caches in WCET Analysis: Predictability – Competitiveness – Sensitivity*. Ph.D. thesis, Saarland University, 2009; <https://bit.ly/3enUrXr>
- Reineke, J. The semantic foundations and a landscape of cache-persistence analyses. *LITES* 5, 1 (2018), 03:1–03:52; doi:10.4230/LITES-v005-i001-a003.
- Reineke, J., Grund, D., Berg, C., and Wilhelm, R. Timing predictability of cache replacement policies. *Real-Time Systems* 37, 2 (2007), 99–122.
- Shaw, A.C. Deterministic timing schema for parallel programs. V.K. Prasanna Kumar, Ed. In *Proceedings of the 5th Intern. Parallel Processing Symp.* (Anaheim, CA, USA, Apr. 30–May 2, 1991), 56–63. IEEE Computer Society; doi:10.1109/IPPS.1991.153757.
- Theiling, H. *Control Flow Graphs for Real-Time Systems Analysis: Reconstruction from Binary Executables and Usage in ILP-based Path Analysis*. Ph.D. thesis, Saarland University, Saarbrücken, Germany, 2003; <http://scidok.sulb.uni-saarland.de/volltexte/2004/297/index.html>.
- Thesing, S. *Safe and Precise WCET Determinations by Abstract Interpretation of Pipeline Models*. Ph.D. thesis, Saarland University, 2004.
- Thesing, S. Modeling a system controller for timing analysis. S.L. Min and W. Yi, Eds. In *Proceedings of the 6th ACM & IEEE Intern. Conference on Embedded Software* (Seoul, Korea, Oct. 22–25, 2006), 292–300; doi:10.1145/1176887.1176929.
- Wilhelm, R., Grund, D., Reineke, J., Schlickling, M., Pister, M., and Ferdinand, C. Memory hierarchies, pipelines, and buses for future architectures in time-critical embedded systems. *IEEE Trans. on CAD of Integrated Circuits and Systems* 28, 7 (2009); doi:10.1109/TCAD.2009.2013287.
- Wilhelm, S. *Symbolic Representations in WCET Analysis*. Ph.D. thesis, Saarland University, 2012; <http://scidok.sulb.uni-saarland.de/volltexte/2012/4914/>.
- Wilhelm, S. and Wachter, B. Symbolic state traversal for WCET analysis. S. Chakraborty and N. Halbwachs, Eds. In *Proceedings of the 9th ACM & IEEE Intern. Conf. Embedded Software*. (Grenoble, France, Oct. 12–16, 2009), 137–146; doi:10.1145/1629335.1629354.

Reinhard Wilhelm (wilhelm@cs.uni-saarland.de) is Professor Emeritus at Saarland University in Saarbrücken, Germany.

Copyright held by author/owner.

CCGrid 2021

21th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing

May 10-13, 2021, Melbourne, Australia

<http://www.cloudbus.org/ccgrid2021>

CALL FOR PAPERS

The 21st IEEE/ACM international Symposium on Cluster, Cloud and Internet Computing (CCGrid 2021) is a leading forum to disseminate and discuss research activities and results on a broad range of topics in distributed systems, ranging from computing Clusters to widely distributed Clouds and emerging Internet computing paradigms such as Fog/Edge Computing for Internet of Things (IoT)/Big Data applications. The conference features keynotes, technical presentations, posters, workshops, tutorials, as well as the SCALE challenge featuring live demonstrations and the IC FEC 2021 conference.

In 2021, IEEE/ACM CCGrid 2021 will be 'self-located' with its postponed 2020 edition, in Melbourne, Australia. We will jointly celebrate the 20th and 21st anniversary of the conference! We solicit original contributions on all aspects of distributed systems and applications in the context of Cluster, Cloud, and Internet computing environments. Specific topics of interest include but are not limited to the following:

- **Internet Computing Frontiers:** Edge, Fog, Serverless, Lambda, Streaming - Highly decentralized approaches to cloud computing. Edge/Fog computing, sensor data streaming and computation on the edges of the network. Function as a Service (Faas), Backend as a Service (BaaS), serverless computing, lambda computing.
- **Architecture, Networking, Data Centers:** Service oriented architectures. Utility computing models. IaaS, PaaS, SaaS, *aaS paradigms. Service composition and orchestration. Software-Defined Network-enabled Systems. Micro-datacenter, cloudlet, edge, or fog computing infrastructure. Virtualized hardware: GPUs, tensor processing units, FPGAs.
- **Storage and I/O Systems:** Distributed storage, cloud storage, Storage as a Service, data locality techniques for in-memory processing, storage in the edge.
- **Programming Models and Runtime Systems:** Programming models, languages, systems and tools/environments. Virtualization, containers, and middleware technologies. Actors, agents, programming decentralized computing systems.
- **Resource Management and Scheduling:** Resource allocation algorithms, profiling, modeling. Cluster, cloud, and internet computing scheduling and meta-scheduling techniques.
- **Performance Modelling and Evaluation:** Performance models. Monitoring and evaluation tools. Analysis of system/application performance.
- **Cyber-Security, Privacy and Resilient Distributed Systems:** Distributed Systems security and trust. Access

control. Data privacy and integrity. Regulation. Resiliency of service attacks.

- **Sustainable and Green Computing:** Environment friendly computing ecosystems. Hardware/software/application energy efficiency. Power, cooling and thermal awareness.
- **Applications:** Data Science, Artificial Intelligence, Machine Learning, Cyber-Physical Systems, e-Health, Internet of Things (IoT)-enabled Smart Systems and Applications.

PAPER SUBMISSION

Authors are invited to submit papers electronically. Submitted manuscripts should be structured as technical papers and may not exceed 10 letter size (8.5 x 11) pages including figures, tables and references using the IEEE format for conference proceedings. All manuscripts will be reviewed and will be judged on correctness, originality, technical strength, significance, quality of presentation, and relevance to the conference attendees.

Submitted papers must represent original unpublished research that is not currently under review for any other conference or journal. Papers not following these guidelines will be rejected without review and further action may be taken, including (but not limited to) notifications sent to the heads of the institutions of the authors and sponsors of the conference. Submissions received after the due date, exceeding length limit, or not appropriately structured may also not be considered. Authors may contact the conference chairs for more information. The proceedings will be published through the IEEE Press, USA and will be made online through the IEEE and ACM Digital Libraries.

CHAIRS & COMMITTEES

General Chair

Rajkumar Buyya, University of Melbourne, Australia
Gul Agha, University of Illinois at Urbana-Champaign, USA

Program Committee Co-Chairs

Laurent Lefevre, INRIA, France
Stacy Patterson, RPI, USA
Young Choon Lee, Macquarie University, Australia

Workshops Co-Chairs

George Pallis, University of Cyprus, Cyprus
Haiying Shen, University of Virginia, USA

IMPORTANT DATES (TENTATIVE)

Papers Due:	December 8, 2020
Notification of Acceptance:	February 8, 2021
Camera Ready Papers Due:	March 3, 2021

Software weaknesses in cryptocurrencies create unique challenges in responsible revelations.

BY RAINER BÖHME, LISA ECKEY, TYLER MOORE, NEHA NARULA, TIM RUFFING, AND AVIV ZOHAR

Responsible Vulnerability Disclosure in Cryptocurrencies

DESPITE THE FOCUS on operating in adversarial environments, cryptocurrencies have suffered a litany of security and privacy problems. Sometimes, these issues are resolved without much fanfare following a disclosure by the individual who found the hole. In other cases, they result in costly losses due to theft, exploits, unauthorized coin creation, and destruction. These experiences provide regular fodder for outrageous news headlines. In this article, we focus on the disclosure process itself, which presents unique challenges compared to other software projects.¹⁵ To illustrate, we examine some recent disclosures and discuss difficulties that have arisen.

While Bitcoin is the best known, more than 2,000 cryptocurrencies are in circulation, collectively valued at \$350 billion as of August 2020.⁶ Figure 1 conceptualizes the landscape as a stack. While the details differ, at the lowest level, each cryptocurrency

system is designed to achieve common security goals: transaction integrity and availability in a highly distributed system whose participants are incentivized to cooperate.³⁸ Users interact with the cryptocurrency system via software “wallets” that manage the cryptographic keys associated with the coins of the user. These wallets can reside on a local client machine or be managed by an online service provider. In these applications, authenticating users and maintaining confidentiality of cryptographic key material are the central security goals. Exchanges facilitate trade between cryptocurrencies and between cryptocurrencies and traditional forms of money. Wallets broadcast cryptocurrency transactions to a network of nodes, which then relay transactions to miners, who in turn validate and group them together into blocks that are appended to the blockchain.

Not all cryptocurrency applications revolve around payments. Some cryptocurrencies, most notably Ethereum, support “smart contracts” in which general-purpose code can be executed with integrity assurances and recorded on the distributed ledger. An explosion of token systems has appeared, in which particular functionality is expressed and run on top of a cryptocurrency.¹² Here, the promise is that business logic can be specified in the smart contract and confidently executed in a distributed fashion.

The emergence of a vibrant ecosystem of decentralized cryptocurrencies has prompted proposals that leverage the underlying technology to construct new central bank currency² and corpo-

» key insights

- **Cryptocurrency software is complex and vulnerabilities can be readily, and anonymously, monetized.**
- **Responsible vulnerability disclosure in cryptocurrencies is hard because decentralized systems, by design, give no single party authority to push code updates.**
- **This review of case studies informs recommendations for preventing catastrophic cryptocurrency failures.**



rate electronic money, such as Facebook’s asset-linked Libra. This article focuses on existing decentralized cryptocurrencies. Some lessons discussed here could also inform the design and operation of these prospective forms of digital money issued by public or private legal entities.

Bugs in cryptocurrencies. The cryptocurrency realm itself is a virtual “wild west,” giving rise to myriad protocols each facing a high risk of bugs. Projects rely on complex distributed systems with deep cryptographic tools, often adopting protocols from the research frontier that have not been widely vetted. They are developed by individuals with varying level of competence (from enthusiastic amateurs to credentialed experts), some of whom have not developed or managed production-quality software before. Fierce competition between projects and companies in this area spurs rapid development, which often pushes developers to skip important steps necessary to secure their codebase. Applications are complex as they require the interaction between multiple software components (for example, wallets, exchanges, mining pools). The high prevalence of bugs is exacerbated by them being so readily monetizable. With market capitalizations often measured in the billions of dollars, exploits that steal coins are simultaneously lucrative to cybercriminals and damaging to users and other stakeholders. Another dimension of importance in cryptocurrencies is the privacy of users, whose transaction data

is potentially viewable on shared ledgers in the blockchain systems on which they transact. Some cryptocurrencies employ advanced cryptographic techniques to protect user privacy, but their added complexity often introduces new flaws that threaten such protections.

Disclosures. Disclosures in cryptocurrencies have occurred in varying circumstances, from accidental discoveries, through analysis by expert developers and academics, to observing successful exploits in the wild. In the rest of this article, we highlight the difficulties and subtleties that arise in each case. The root causes of most of the difficulties lie in the special nature of cryptocurrencies: they are based on distributed systems that were designed to be difficult to change in order to provide strong guarantees on their future behavior. In order to change these rules, the consent of many participants is needed—participants who are often anonymous, and who are organized loosely in communities without governing bodies or regulatory oversight.

Here, we briefly highlight the differences between conventional software development and cryptocurrencies with regard to vulnerability disclosure, we identify key issues in the disclosure process for cryptocurrency systems, and we formulate recommendations and pose open questions.

How Is Disclosure Different?

Responsible vulnerability disclosure in cryptocurrencies differs from the con-

ventions adopted for general software products in several ways. Two fundamental differences arise from the very nature of cryptocurrencies.

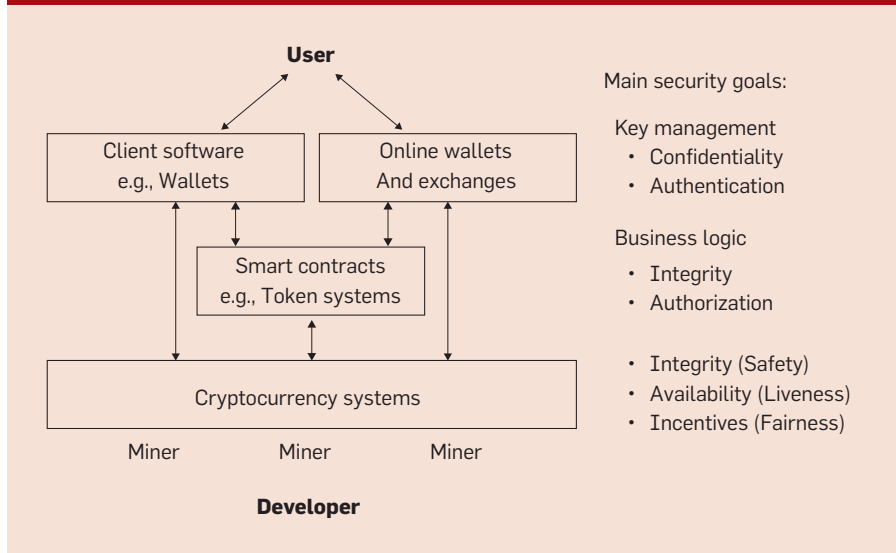
First, the decentralized nature of cryptocurrencies, which must continuously reach system-wide consensus on a single history of valid transactions, demands coordination among a large majority of the ecosystem. While an individual can unilaterally decide whether and how to apply patches to her client software, the safe activation of a patch that changes the rules for validating transactions requires the participation of a large majority of system clients. Absent coordination, users who apply patches risk having their transactions ignored by the unpatched majority.

Consequently, design decisions such as which protocol to implement or how to fix a vulnerability must get support from most stakeholders to take effect. Yet no developer or maintainer naturally holds the role of coordinating bug fixing, let alone commands the authority to roll out updates against the will of other participants. Instead, loosely defined groups of maintainers usually assume this role informally.

This coordination challenge is aggravated by the fact that unlike “creative” competition often observed in the open source community (for example, Emacs versus vi), competition between cryptocurrency projects is often hostile. Presumably, this can be explained by the direct and measurable connection to the supporters’ financial wealth and the often minor technical differences between coins. The latter is a result of widespread code reuse,²⁸ which puts disclosers into the delicate position of deciding which among many competing projects to inform responsibly. Due to the lack of formally defined roles and responsibilities, it is moreover often difficult to identify who to notify within each project. Furthermore, even once a disclosure is made, one cannot assume the receiving side will act responsibly: information about vulnerabilities has reportedly been used to attack competing projects,¹⁸ influence investors, and can even be used by maintainers against their own users.

The second fundamental difference emerges from the widespread design goal of “code is law,” that is, making code the final authority over the shared

Figure 1. Components of the cryptocurrency architecture covered in this article.



system state in order to avoid (presumably fallible) human intervention. To proponents, this approach should eliminate ambiguity about intention, but it inherently assumes bug-free code. When bugs are inevitably found, fixing them (or not) almost guarantees at least someone will be unhappy with the resolution. This is perhaps best exemplified by the controversy around the DAO, an Ethereum smart contract with a reentrance bug that was exploited to steal coins worth around \$50 million. After a community vote, the Ethereum developers rolled out a patch to reverse the heist, which (maybe surprisingly) turned out to be controversial. While the patch was accepted by large parts of the ecosystem, it was strongly opposed by a minority of Ethereum users arguing that it is a direct violation of the code-is-law principle, and the controversy ultimately led to a split of the Ethereum system into two distinct cryptocurrencies Ethereum and Ethereum Classic.¹ Moreover, situations may arise where it is impossible to fix a bug without losing system state, possibly resulting in the loss of users' account balances and consequently their coins. For example, if a weakness is discovered that allows anybody to efficiently compute private keys from data published on the blockchain,¹⁶ recovery becomes a race to move to new keys because the system can no longer tell authorized users and attackers apart. This is a particularly harmful consequence of building a system on cryptography without any safety net. The safer approach, taken by most commercial applications of cryptography but rejected in cryptocurrencies, places a third party in charge of resetting credentials or suspending the use of known weak credentials.

Ironically, these fundamental differences stem from design decisions intended to enhance security. Decentralization is prized for eliminating single points of control, which could turn out to be single points of failure. Giving code the final say is intended to preserve the integrity of operations. However, what may benefit security at design time becomes a significant liability after deployment once vulnerabilities are found.

Besides these fundamental differences, responsible disclosure for cryp-



The decentralized nature of cryptocurrencies, which must continuously reach system-wide consensus on a single history of valid transactions, demands coordination among a large majority of the ecosystem.



tocurrencies is characterized by specific features of the domain. The interpretation of system state as money, with many exchanges linking it mechanically to the conventional financial system, makes it easier and faster to monetize bugs than for conventional software, where vulnerability markets may exist but are known to be friction-prone.²³ Moreover, the cryptocurrency ecosystem reflects conflicting world-views, which prevent the establishment of basic norms of acceptable behavior. For example, invalidating ransomware payments via blacklisting has reignited the debate over censorship and the rule of law.²⁶

Finally, we note a difference in emphasis over certain aspects of disclosure. The conventional responsible disclosure discussion has focused on balancing users' interests in defensively patching versus national security interests of weaponizing vulnerabilities,^{25,31} without regard to whether the affected software is open or closed source. By contrast, open source software and code reuse are central to disclosure issues in cryptocurrencies, whereas balancing national and individual security considerations has so far not been widely discussed.

Throughout the rest of the article, we illustrate these differences with real cases before we derive recommendations and point to open problems.

Case Studies

We now review selected case studies of cryptocurrency vulnerability disclosures, highlighting aspects that teach us about the difficulties in response. We employ a multi-perspective method in selecting and researching these cases, ranging from the authors' direct experience as disclosers, interviews with developers and cryptocurrency designers, and through public reports. Interviews with open-ended questions were conducted by telephone, in-person or by email. Attribution is given unless the subject requested anonymity. The novelty and heterogeneity of the problem precluded a more systematic approach, though we hope that those informed by our findings can do so in future investigations. We investigate coins both small and large, because even the top coins have experienced severe bugs. While the software development pro-

cesses for prominent coins are more robust, the cases will show that all coins experience challenges to disclosure not seen in traditional software projects. Figure 2 presents a stylized timeline of the cases presented.

Cryptocurrency systems. Zcoin. We start with Zcoin, a relatively little known cryptocurrency that has suffered from repeated disclosures. Zcoin was the first to implement the Zerocoin protocol,²² which uses zero-knowledge proofs to enable untraceable transactions. In February 2017, an attacker exploited a typo in C++ code¹⁷ (using the equality operator ‘==’ instead of the assignment operator ‘=’) to generate 403,050 coins out of thin air. The new coins had a market value of \$750,000 and inflated the currency supply in cir-

ulation by 37%. In principle, such attacks can remain unnoticed due to the zero-knowledge veil, but the sheer number of coins created combined with the attacker’s impatience eventually led to its discovery. Within hours, the Zcoin team demanded that trading halt at big exchanges, published a blog post, and asked mining pools to suspend processing zero-knowledge transactions. A patch was released within a day, but the zero-knowledge feature remained disabled, thereby temporarily freezing all untraceable funds. This issue was resolved after four days when a “fork” altering the fundamental transaction validation rules was adopted by a majority of the miners. Even so, the attacker was able to abscond with the loot.

Later in 2017, a team of researchers including author Ruffing found another vulnerability in Zcoin that allowed an attacker to “burn” money in transit, that is, ensure no one, including the sender, recipient, and attacker, can further spend the coins.³⁰ Remarkably, the root cause of this vulnerability was an overlooked attack vector in the design and security analysis of the underlying Zerocoin protocol. While money burning does not serve the attacker directly, the attacker could profit indirectly, for example, by betting on falling prices of the affected cryptocurrency (short selling) and then publishing or exploiting the vulnerability. We have no evidence that such short-selling activity did indeed take place.

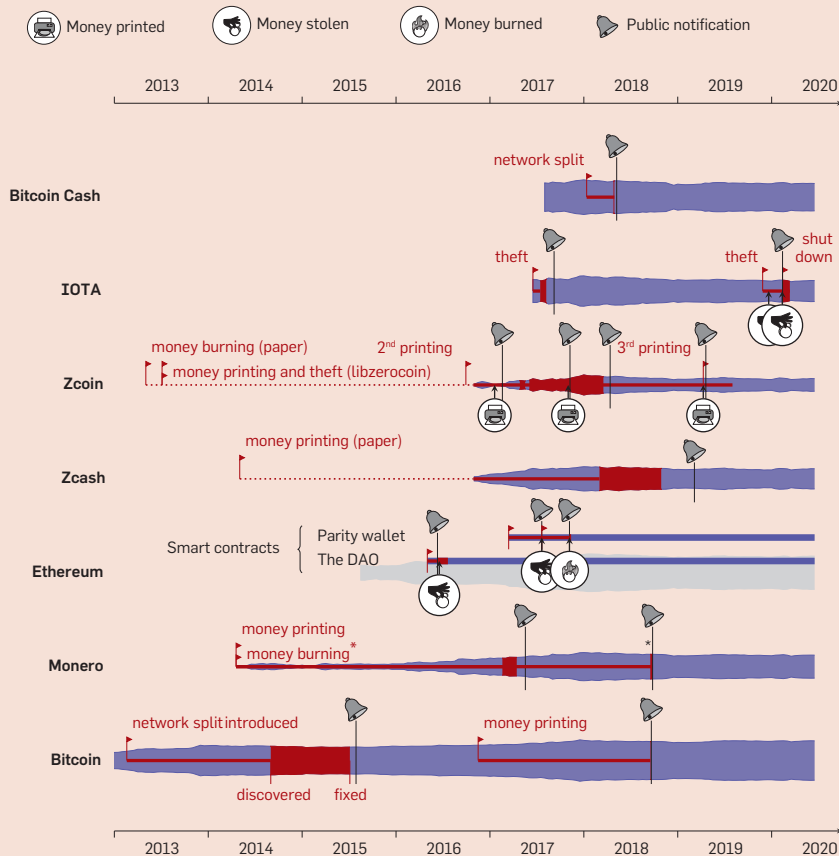
Having no cryptographer on its team, Zcoin hired Ruffing to provide advice and develop a patch. During the work, he identified two more vulnerabilities,²⁹ one enabling illegitimate coin generation and one allowing theft of money in transit. Both vulnerabilities stemmed from bugs in libzerocoin, a prototype library written by the inventors of the Zerocoin protocol for the purpose of validating their research. The Zcoin project had used that library as-is, despite the code’s prominent warning that the authors “are releasing this dev version for the community to examine, test and (probably) break” and that there are things that they have “inevitably done wrong.”²¹

Code reuse complicated the disclosure process of the three vulnerabilities.²⁹ Months after the initial notification, the discoverers found that more than 1,600 public GitHub repositories included verbatim copies of libzerocoin. Responsible and confidential disclosure to so many recipients is infeasible. Instead, the discoverers narrowed down the recipient set to less than 10 actual cryptocurrency projects, four of which they deemed trustworthy enough to be informed additionally. None of the projects had a clearly defined contact point or process for handling vulnerabilities.

Competition between projects prevented a coordinated response. For example, the notified project did not reveal to the reporters which of their competitors were also vulnerable. Coordination is essential because the first project to patch reveals the vulnerabili-

Figure 2. Visualization of the vulnerabilities discussed in this article.

The blue bars represent the underlying coins and their widths are proportional to their marketcap (for example, Coinmarketcap.org). The red bars visualize the discussed incidents from their introduction (flag) to their disclosure (wide bar) to their public announcement (bell). The additional symbol is used whenever money was stolen, burnt or printed.



ty, leaving the others unprotected. One currency was actually exploited in this way, and ironically, Zcoin itself was targeted because the patch was not adopted quickly enough. Dealing with the entire situation required tact and judgment by the discoverers, and the potential for every mistake to be catastrophic furthers the discoverers' burden.

As a result of the coin creation bugs, Zcoin improved continuous monitoring of aggregated balances, which led to the discovery of another creation bug in April 2019. The project repeated the notification process described earlier, disabled the zero-knowledge features via an emergency fork, and informed three potentially affected competitors. It took 10 days of investigation before a project developer identified the root cause in the design of the Zerocoin protocol. Unlike a simple implementation bug, there was no obvious way to fix the problem. The project's response was to migrate to an entirely different zero-knowledge protocol, suspending untraceable transactions in the meantime and freezing the affected funds until the new protocol was deployed in July 2019.³⁷

Zcash. Zcash, the commercial implementation of the Zerocash protocol,³ improves on Zerocoin's model for untraceable transactions. It too has suffered from similar issues.³² The proposal for the used algorithm for generating cryptographic material allowed a parameter to be published that should have remained secret. (Incidentally, a security proof was omitted because the scheme was similar to a previous one known to be secure.) The published value could have been used to undetectably generate coins out of thin air. The problem was discovered internally in March 2018 and fixed after 240 days in conjunction with a scheduled upgrade of the zero-knowledge protocol. Before and during the events the Zcash team had entered mutual disclosure agreements with the two largest competitors who reuse Zcash code. These competitors were notified two weeks after the fix with a schedule for public disclosure within a maximum of 90 days, which then took place in February 2019, almost one year after the discovery.³² Obscurity played a key role in this event: not only was the fix hidden in a larger update, the critical parameter was also



Unlike bugs in which coins are created, IOTA suffered a vulnerability that might have placed user funds at risk of theft.



removed from websites and a cover story spun around the "loss" of this piece of information. The intention of this obscurity was to protect Zcash's own interests and its users, as well as those of competing cryptocurrencies. On the downside, such long periods of obscurity may cast doubt on the trustworthiness of security claims in the future, and it remains unclear whether and to what extent the bug has been exploited.

Monero. The opposite of internal discovery is accidental public disclosure. This happened to Monero, the most popular implementation of the CryptoNote protocol.³⁵ In September 2018, an interested user posted a seemingly innocuous question to an online forum: "What happens if somebody uses a one-time account twice?" (paraphrased by the authors).⁷ Surprisingly, there was no protection against this action in the protocol. The revealed vulnerability allowed attackers to burn other people's funds. The problem was fixed within 10 days without known incidents and publicly announced thereafter.

A more serious vulnerability in the CryptoNote protocol affected all cryptocurrencies based on it. A post on a specialized cryptography mailing list in February 2017 revealed an issue, which implied a coin generation vulnerability in CryptoNote's basic cryptographic scheme.²⁰ The Monero team took note and developed a patch within three days and shared it privately with preferred parties, such as mining pools and exchanges. The true purpose of the patch was disguised in order to protect the rest of the users who were running vulnerable clients. After a fork to the validation rules that completely resolved the issue in Monero in April 2017, the Monero team informed other CryptoNote coins privately. One such coin, Bytecoin, was exploited immediately afterward, resulting in the illegitimate generation of 693 million coins.¹⁸ In a public disclosure that took place 15 days later, the Monero team described the aforementioned process and named unpatched competitors, including Bytecoin²⁰ (though Bytecoin claims that a patch had been issued to miners immediately after the exploit¹⁸). Perversely, the public disclosure attracted other investors to bid up the Bytecoin price. Its market capitalization grew

five-fold, briefly jumping into the top 10 cryptocurrencies by value. It remains unclear who exploited the bug, but Bitcoin holders certainly benefited from the price rise.

IOTA. Unlike bugs in which coins are created, IOTA suffered a vulnerability that might have placed user funds at risk of theft. Contrary to the best practice of using standardized cryptographic primitives, IOTA relied on a custom hash function that had a collision weakness.¹⁴ Author Narula and colleagues disclosed the vulnerability to the developers in July 2017. The vulnerability was patched by IOTA in August 2017 and made public by the disclosers in September 2017,¹³ offering several lessons about the disclosure process.

First, the vulnerability was fixed and deployed to the network quite quickly. On one hand, this is good because the potential vulnerability window is smaller. On the other hand, the speedy response was made possible due to the project's high level of control over the network, which runs contrary to the design goals of decentralized cryptocurrencies. Such control further allowed the operators to shut down its network to prevent theft from a vulnerable wallet for several weeks in early 2020.

The second lesson is that organizations may not respond favorably to a disclosure. Here, communications were tense, the existence and risk of the vulnerability was denied and downplayed, and the discoverers were threatened with lawsuits. The response echoes industry reactions to vulnerability disclosures related to digital rights management decades before.¹⁹ In the cryptocurrency case, there is a clear potential incentive conflict when the organization holds a large share of the coins and reasonably worries that the news could devalue holdings or prevent partnerships that might increase the value of holdings. Moreover, information about the bug could be exploited for profit by those possessing inside information about its existence prior to public disclosure.

Bitcoin Cash. Not to be confused with Bitcoin, "Bitcoin Cash" is derived from Bitcoin's codebase and was created due to disagreements within the ecosystem. Cory Fields, a contributor to the predominant implementation of Bitcoin, Bitcoin Core, was examining

change-logs of Bitcoin Cash's main implementation in April 2018.¹⁰ There he noticed that a sensitive piece of code dealing with transaction validation had been improperly refactored, causing a vulnerability. It would allow an attacker to split the Bitcoin Cash network, thereby compromising the consistency required for a cryptocurrency to operate.

As Fields noted, bugs like this cause systemic risk: if exploited, they could sink a cryptocurrency. The large amounts of money at risk prompt disclosers to take precautions. In this case, to protect his own safety, Fields chose to remain anonymous.¹⁰ The patching went smoothly, but we do not know if it would have been more contentious had he revealed his identity. Moreover, discoverers may want to demonstrate they behaved ethically, for example, that they sent a report to the developers. One possible mechanism is to encrypt the report with the developers' public key and publish the ciphertext and draw the developer's attention to it. This would require developers to provide public keys along with their security contact and have internal processes to handle incoming messages. Surprisingly, at the time Bitcoin Cash, a top-10 cryptocurrency worth billions of dollars, did not (though now they do). In our interview, Fields stressed he found it difficult to figure out what was the right thing to do. What helped him was to imagine the situation with swapped roles.

Bitcoin. A few months later, a developer from Bitcoin Cash disclosed a bug to Bitcoin (and other projects) anonymously. Prior to the Bitcoin Cash schism, an efficiency optimization in the Bitcoin codebase mistakenly dropped a necessary check. There were actually two issues: a denial-of-service bug and potential money creation.⁸ It was propagated into numerous cryptocurrencies and resided there for almost two years but was never exploited in Bitcoin.

This case teaches us three lessons: First, even the most watched cryptocurrencies are not exempt from critical bugs. Second, not all cases should be communicated to everyone in the network at the same time. The Bitcoin developers notified the miners controlling the majority of Bitcoin's hashrate of the denial-of-service bug first, making sure they had upgraded so that nei-

ther bug could be exploited before making the disclosure public on the bitcoin-dev mailing list. They did not notify anyone of the inflation bug until the network had been upgraded. Third, the disclosure involved deliberate deception of users: the Bitcoin developers published a patch describing it as only fixing the denial-of-service issue. This downplayed the severity of the bug, while at the same time motivating a prompt upgrade. This gave Bitcoin users and other affected cryptocurrencies time to adopt the fix, albeit with grumbling about the sudden public release. This highlights both a benefit and a downside to employing white lies in the disclosure process.

Silence is an alternative to white lies. The Bitcoin team took this option after an internal discovery in 2014. Bitcoin suffered from an inconsistency between different versions of the OpenSSL library. The 32-bit version was more tolerant in accepting variants of digital signatures than the 64-bit version, which could cause a loss of consistency if a signature is accepted only by the subset of nodes running on 32-bit. The mitigation turned into a year-long ordeal. Fixing OpenSSL was not an option, hence the stricter signature format had to be enforced in the Bitcoin codebase. Changes were made subtly and gradually in order to avoid drawing attention on the relevant piece of code. Users upgraded organically over a period of 10 months. The bug was made public when more than 95% of the miners had patched.³⁶

Smart contracts. Some cryptocurrencies, most prominently Ethereum, support "smart contracts." These are computer programs anyone can store on a shared blockchain, which then purports to guarantee correct execution. Contracts can receive, store, and send coins to users or other contracts according to their programmed logic. Smart contracts pose two further challenges to disclosure and patching. First, there is no club of miners whose incentives are aligned with the functioning of a specific contract. Therefore, relying on miners as allies to support smooth disclosure is usually not an option (though we will discuss an exception). Second, the code is not updatable by design to demonstrate commitment to the rules of operation,

hence the contract analogy. This may turn disastrous if the code contains bugs because machines, unlike arbitrators of real contracts, have no room for interpretation.

The DAO. The most famous example of a buggy contract is the DAO (short for Decentralized Autonomous Organization), the first code-controlled venture fund. Widely endorsed by an enthusiastic Ethereum community, in spring 2016 the DAO project collected user funds and stored them in a smart contract. Its visible balance of \$250 million (15% of all available coins at the time) made it a highly attractive target. It prompted scrutiny from security researchers who raised concerns,⁹ the closest activity to disclosure in the smart contract space we have seen. Three weeks later, an anonymous attacker managed to withdraw more than 3.5 million coins (about \$50 million) illegitimately from the DAO smart contract.¹ The attacker's trick involved making a small investment in the DAO, then withdrawing and thereby exploiting a re-entrance vulnerability in the refund mechanism. (The contract's bug was to not decrease the balance before sending coins, which in Ethereum passes control to the receiving party.) This exploit set off a vigorous debate over whether or not this behavior was abusive, since the code technically allowed the interaction.

The DAO incident could have been an example of an irreversible change of system state. However, the exceptional scale of the project and the involvement of the Ethereum community triggered a historic vote between miners to support a fork of the underlying cryptocurrency in order to “restore” the investments in the DAO contract. This intervention was highly controversial as it thwarted the very idea of immutable transactions, causing a group of purists to create a parallel instance, called Ethereum Classic, that was not rolled back. In hindsight, the incident raised the alarm to the smart contract community about the looming security issues. Today's contracts cannot hope for miner-enforced rollbacks because the uptake of the platform has diversified interests.

Parity wallet. Another example of a fund recovery, albeit partially successful, followed the Parity exploit in July

2017. The vulnerable contract implemented a multi-signature wallet, a mechanism that promises superior protection against theft compared to standard wallets. Intended uses include “corporate” accounts storing high value, such as the proceeds from initial coin offerings (ICOs). An anonymous attacker observed a discrepancy between the published and reviewed source code and the binary code, which was deployed for each of 573 wallets and omitted an essential access control step. This enabled a theft of coins worth \$30 million from three accounts. Parity discovered the attack as it was ongoing and published an alert. This would have enabled attentive users to rescue their funds (exploiting the same vulnerability) in a race against the attacker and imitators. At this point, a total of another \$150 million was essentially free to be picked up by anyone.³³ As expected, many users reacted slowly and found their funds missing. It turned out that a group of civic-minded individuals has taken the funds in custody in order to protect users and return them in a safe way. This example raises the question if protective appropriation of funds is legal or should even be expected from discoverers.

Users who nevertheless continued to trust the Parity wallet software were less lucky following a second incident. The Ethereum platform has a fuse mechanism that irrevocably disables code at a given address. In November 2017, a user (allegedly) inadvertently invoked this mechanism on a library referenced in 584 intentionally non-updatable contracts of the next-generation Parity wallet. A total of \$152 million was burned.³⁴ This time, no one intervened, presumably because the loss concerned only 0.5% of all coins.

We close by noting that as of this writing, we are not aware of any major

cases of responsible disclosures of vulnerabilities in smart contracts.

Recommendations and Open Questions

While best practices in secure software engineering and responsible disclosure¹⁵ are increasingly adopted in the cryptocurrency space, there always remains a residual risk of damaging vulnerabilities. Therefore, norms and eventually laws for responsible disclosure must emerge. What follows is a first step toward that end. Our synthesis of what can be learned from the cases is structured along three central issues of responsible disclosure: how to protect users, who to contact, when and how; and, how to reward the discoverer. The accompanying table sums up the recommendations outlined in this section.

How to protect users. *Discoverer safety.* If the vulnerability can make parties who may operate beyond the law substantially richer or poorer, the discoverer's personal safety should be considered.¹⁰ Death threats are not unheard of. Confidentially sharing the vulnerability with others the discoverer trusts (professional colleagues, notaries or the police) might reduce this risk. Sealed envelopes, or their digital variants such as time-locked encryption or secret sharing schemes, lessen the risk of unintended leakage. In addition, anonymous reporting may also reduce stress and tension. However, note that if the vulnerability is exploited, any proof the discloser knew of the vulnerability before its exploit could be used as evidence the discloser was the attacker.

Addressing vulnerable funds. If a vulnerability means that anyone can steal money from an account, should civic-minded defenders proactively steal to protect funds, like in the Parity wallet case? This touches on unresolved legal

Synthesis of recommendations.

Do's	Provide point of contact including public key Liaise with competitors who share code
Don'ts	Single out vulnerable competitors Bug bounties in your own coin
Depends	Use obscurity and white lies during disclosure Notify all affected projects unless there is conflict Built-in notification and feature “kill” switches
Need for action	Clarify right or obligation to preventively move vulnerable funds Establish clearinghouse and coordinator

questions. If “code is law” is the guiding principle, moving vulnerable funds must be legal. But courts are bound to real-world norms which differ across jurisdictions and circumstances. For example, in many places only law enforcement can legally expropriate property, including crypto coins. Elsewhere, disclosers could be obligated to intervene rather than stand by and allow a crime to take place. To give the discoverer legal certainty, it is essential to settle the basic question whether the discoverer could face legal consequences if she takes such precautions or break the law if she has the power and does not. If opting to leave the matter to law enforcement, other complications arise: Which law enforcement agency has jurisdiction and sufficient authority and is allowed to act? Do all law enforcement agencies possess the technical capability to intervene in time?

Preparing the system for disclosure. Given the inevitability of vulnerabilities, one strategy is to implement features in the cryptocurrency itself to automatically notify affected users of significant problems. In fact, Bitcoin used to have such an alert system, which enabled trusted actors to disseminate messages to all users and even suspend transactions. Such alert systems prompt difficult questions of their own, like who can be trusted with that authority in a decentralized system? Also, the alert system itself could become the target of attack, in much the same way that an Internet “kill switch” could create more security problems than it solves. Incidentally, Bitcoin itself abandoned the alert system over such concerns.⁴ A similar idea is to incorporate a mechanism to turn off particular features if significant vulnerabilities are later found. Dash utilizes such a system that lets the holder of a secret key turn features on and off at will.²⁷ PIVX supports a similar mechanism to disable zero-knowledge transactions, which proved useful during the Zerocoin disasters.

Despite the benefits such features bring, they contradict the design philosophy of decentralization and might expose the privileged party to law enforcement requests. Supposing a cryptocurrency could overcome these challenges and develop mechanisms for disseminating protective instructions,

the question of how to contact the trusted party who takes the precaution remains.

Who to contact, when, and how. *Provide clear points of contact.* Many cryptocurrencies are designed to avoid relying on privileged parties with substantial control. Yet this is in effect required to support responsible disclosure. It can be difficult to determine who is “in charge” (assuming anyone is) and who can fix the bug. Best practices recommend that developers provide clear points of contact for reporting security bugs, including long-term public keys.¹¹ Developers who reuse code are advised to publish alongside their own contact information that of the original code to aid the search for affected projects.

Identifying the responder. All communication by the discoverer should serve the end of fixing the bug. This means the discoverer must notify the party who is in the best position to solve the problem. For example, if the vulnerability affects the cryptocurrency’s core implementation, then the developers are the natural responders. There is a long history of bugs in exchanges,²⁴ in which case they would respond. It is important to note that once the responder has taken responsibility, the discoverer should adopt a “need-to-know” practice until the risk is mitigated. Sometimes the natural choice for responder is missing or untrustworthy. In this case, the discoverer can also serve as responder, or delegate the responsibility to a third party.

Responder communication with stakeholders. Given the decentralized nature of cryptocurrencies, the responder is usually not in a position to unilaterally act to fix the bug. Instead, the responder must seek stakeholders’ support. This means communicating the right messages at the right time. It could be dangerous to tell the full truth right away, so the message may justifiably include obfuscation or even white lies. Different stakeholders might require varying levels of detail at particular points in time. For bugs that require certain transactions to be mined for successful exploitation, the responder might encourage miners to upgrade first in order to deploy a fix as fast as possible. Exchanges can suspend trading in order to limit price shocks as bad news breaks, or aid in blocking the

transport of stolen funds. In other instances, wallet developers must be notified first, in order to deploy patches to their software. It is good practice to publish an advisory detailing the course of events and clarify any obfuscation or lies after the risk is mitigated. This transparency could mitigate the erosion of trust resulting from deception.

Coordination among multiple responders. As illustrated in the cases explored here, vulnerabilities often affect multiple projects. It is up to the discoverer to decide where to send the report. The reporter should be transparent about who has been informed. The discoverer can work with the responders to ensure that everyone affected has been notified. Coordination among responders is essential. Patches should be deployed as simultaneously as possible across affected projects, since the patching and publication of vulnerability information would leave others exposed if no precautions were taken. In some circumstances, the responders are competitors, and their attitudes toward one another range from suspicion to hostility.

Dealing with untrustworthy responders. While in the traditional security world, it is considered not only common courtesy but professionally and ethically required to inform other projects about vulnerabilities before disclosing their existence publicly. In the cryptocurrency world, one must adopt a more adversarial mindset. If the discoverer does not find a trustworthy responder, she can take on that responsibility. While one might not expect the discoverer to fix the bug, she could nonetheless take steps to protect users.

The situation is further complicated when multiple projects share a problem, and some are not trustworthy or are hostile toward each other. It is unreasonably burdensome for a discoverer to adjudicate such conflicts. Responders can make a best effort to identify affected parties (for example, searching for coins sharing common codebases) and notify accordingly. This points to the need for developing a clearinghouse, à la CERT/CC.


External authorities. Banks, payment processors, and other key financial institutions are often required to report vulnerabilities to banking regulators, who can coordinate the response if

needed. There is no current equivalent for cryptocurrencies, and it is unclear under which jurisdiction such a thing would reside. Should some global reporting agency of this nature be formed? If so, how might it successfully operate given a community whose common ground is removing the need for central parties? An external body modeled on CERT/CC might serve as a useful starting point. A less formal and more decentralized example to consider is iamthecalvary.org, an initiative bringing together security researchers with medical device manufacturers to promote responsible vulnerability disclosure and remediation.

How to reward the discoverer. The article has shown that disclosing a cryptocurrency vulnerability and reacting responsibly is very burdensome. Interviewees have reported sleepless nights and fears for their safety, which in turn has altered their professional collaborations and friendships. The alternative to profit from the vulnerability, potentially anonymously, is tempting. This is why cryptocurrencies specifically cannot expect altruistic behavior and must instead incentivize responsible disclosure.¹¹

Bug bounties offer an established way to reward those who find bugs.⁵ It stands to reason they would be a natural fit for cryptocurrencies, given they have a built-in payment mechanism. However, denominating the reward in its own currency is problematic, since its value might diminish as a result of disclosing the vulnerability, and you are effectively rewarding the discloser in a currency which she just found to be buggy. Other approaches are possible—for example, Augur (a smart contract market platform) is experimenting with exploit derivatives. It is not unreasonable to think that the cryptocurrency community might innovate a solution that could be a model for the broader software community. Nevertheless, monetary rewards must complement and cannot substitute for healthy norms and a culture that welcomes vulnerability disclosure.

Acknowledgments. This article is a result of a breakout group at the Dagstuhl Seminar 18461, “Blockchain Security at Scale.” The authors thank C. Fields, M. Fröwis, P. van Oorschot, and their interview partners.

This work is partially funded by: Archimedes Privatstiftung, Innsbruck, U.S. National Science Foundation Award No. ~ 1714291, ISF grant 1504/17, HUJI Cyber Security Research Center Grant, DFG grants FA 1320/1-1 (Emmy Noether Program) and SFB 1119—236615297 (Crossing), and BMBF grant 16KIS0902 (iBlockchain). 

References

- Atzei, N., Bartoletti, M. and Cimoli, T. A survey of attacks on Ethereum smart contracts. In *Proceedings of the Principles of Security and Trust*. M. Maffei and M. Ryan, Eds. LNCS 10204 (2017), Springer, 164–186.
- Bech, M. and Garratt, R. Central bank cryptocurrencies. *BIS Quarterly Rev.* 9 (2017), 55–70.
- Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I. and Virza, M. Zerocash: Decentralized anonymous payments from Bitcoin. In *Proceedings of the IEEE Symp. on Security and Privacy*, 2014.
- Bishop, B. Alert key disclosure. Bitcoin development mailing list; <https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2018-July/016189.html>.
- Böhme, R. A comparison of market approaches to software vulnerability disclosure. *Emerging Trends in Information and Communication Security*. G. Müller, Ed. LNCS 3995 (2006). Springer, Berlin Heidelberg, 298–311.
- CoinMarketCap. Global charts, 2019; <https://coinmarketcap.com/charts/>.
- dEBRUYN. A post mortem of the Burning Bug, 2018; <https://web.getmonero.org/2018/09/25/a-post-mortem-of-the-burning-bug.html>.
- Bitcoin Core Developers. CVE-2018-17144 Full Disclosure; <https://bitcoincore.org/en/2018/09/20/notice/>.
- Dino, M., Zamfir, V. and Sirer, G.E. A call for a temporary moratorium on the DAO. Blog post; <http://hackingdistributed.com/2016/05/27/dao-call-for-moratorium/>.
- Fields, C. Responsible disclosure in the era of cryptocurrencies. Blog post, 2018; <https://bit.ly/3cgSdYs>.
- Fields, C. and Narula, N. Reducing the risk of catastrophic cryptocurrency bugs. Blog post, 2018; <https://bit.ly/2SKbd9Y>.
- Fröwis, M., Fuchs, A. and Böhme, R. Detecting token systems on Ethereum. In *Proceedings of the Financial Cryptography and Data Security*. I. Goldberg and T. Moore, Eds. 2019.
- Heilman, E., Narula, N., Dryja, T. and Virza, M. IOTA vulnerability report: Cryptanalysis of the curl hash function enabling practical signature forgery attacks on the IOTA cryptocurrency, 2017; <https://github.com/mit-dci/tangled-curl/blob/master/vuln-iota.md>.
- Heilman, E., Narula, N., Tanzer, G., Lovejoy, J., Colavita, M., Virza, M. and Dryja, T. Cryptanalysis of curl-p and other attacks on the IOTA cryptocurrency. IACR Cryptology ePrint archive report 2019/344; <https://eprint.iacr.org/2019/344>.
- Householder, A., Wassermann, G., Manion, A. and King, C. The CERT Guide to Coordinated Vulnerability Disclosure. Special Report CMU/SEI-2017-SR-022.
- Hutchinson, L. All Android-created Bitcoin wallets vulnerable to theft. *Ars Technica*, 2018; <https://bit.ly/2SMHiy5/>.
- Insom, P. Zcoin's Zerocoin bug explained in detail. Blog post, 2017; <https://zcoin.io/zcoins-zerocoin-bug-explained-in-detail/>.
- Juarez, A.M. Fraudulent transactions allowed by the CryptoNote key image bug remain valid. Archived version of Bytecoin GitHub; <https://bit.ly/2WbYkrn>.
- Lok, C. Dispute over digital music muzzles academic. *Nature* 411, 6833 (2001), 5.
- Luigi1111 and Riccardo “fluffypony” Spagni. Disclosure of a major bug in CryptoNote based currencies. Blog post, 2017; <https://bit.ly/2xHiTmb>.
- Miers, I. README of libzerocoin, 2013; <https://bit.ly/2L94ORJ>.
- Miers, I., Garman, C., Green, M. and Rubin, A. Zerocoin: Anonymous distributed e-cash from Bitcoin. In *Proceedings of the 2013 IEEE Symp. on Security and Privacy*.
- Miller, D. The legitimate vulnerability market: Inside the secretive world of 0-day exploit sales. In *Proceedings of the Workshop on the Economics of Information Security*. Carnegie Mellon University, Pittsburgh, PA, 2007.
- Moore, T., Christin, N. and Szurdi, J. Revisiting the risks of Bitcoin currency exchange closure. *ACM Trans. Internet Technology* 18, 4 (2018), 50:1–50:18.
- Moore, T., Friedman, A. and Procaccia, A. Would a ‘cyber warrior’ protect us: Exploring trade-offs between attack and defense of information systems. In *Proceedings of the New Security Paradigms Workshop*. A.D. Keromytis, S. Peisert, R. Ford and C. Gates, Eds. ACM, 2010, 85–94; <https://tylermoore.utulsa.edu/nspw10.pdf>.
- Möser, M. and Narayanan, A. Effective cryptocurrency regulation through blacklisting, 2019; <https://maltemoeser.de/paper/blacklisting-regulation.pdf>.
- Multi-Phased Fork. Glossary item in developer documentation. Dash project, 2017; <https://dash-docs.github.io/en/glossary/spork>.
- Reibel, P., Yousaf, H. and Meiklejohn, S. An exploration of code diversity in the cryptocurrency landscape. In *Proceedings of the 2019 Financial Cryptography and Data Security Conf.* I. Goldberg and T. Moore, eds.
- Ruffing, T., Thyagarajan, S., Ronge, V. and Schröder, D. A cryptographic flaw in Zerocoin (and two critical coding issues). Blog post, 2018; <https://bit.ly/2WAib2B>.
- Ruffing, T., Thyagarajan, S., Ronge, V. and Schröder, D. (Short Paper) Burning Zerocoins for fun and for profit—A cryptographic denial-of-spending attack on the Zerocoin protocol. In *Proceedings of the Crypto Valley Conf. on Blockchain Technology*, (Zug, Switzerland, June 20–22, 2018). IEEE, 116–119; <https://doi.org/10.1109/CVCBT.2018.00023>.
- Schwartz, A. and Knake, R. Government's Role in Vulnerability Disclosure. Harvard Kennedy School discussion paper, 2016.
- Swihart, J., Winston, B., and Bowe, S. Zcash counterfeiting vulnerability successfully remediated. Blog post, 2019; <https://bit.ly/2YD0790/>.
- Parity Technologies. The multi-sig hack: A postmortem, 2017; <https://www.parity.io/the-multi-sig-hack-a-postmortem/>.
- Parity Technologies. A postmortem on the parity multi-sig library self-destruct, 2017; <https://www.parity.io/a-postmortem-on-the-parity-multi-sig-library-self-destruct/>.
- van Saberhagen, N. CryptoNote v 2.0. White Paper, 2013; <https://cryptonote.org/whitepaper.pdf>.
- Wuille, P. Disclosure: Consensus bug indirectly solved by BIP66. Bitcoin development mailing list, 2015; <https://bit.ly/2zeC5rX>.
- Yap, R. Further disclosure on Zerocoin vulnerability. Blog post, 2019; <https://zcoin.io/further-disclosure-on-zerocoin-vulnerability/>.
- Zohar, A. Bitcoin: Under the hood. *Commun. ACM* 58, 9 (Sept. 2015), 104–113.

Rainer Böhme (rainer.boehme@uibk.ac.at) is a professor of computer science at Universität Innsbruck, Austria.

Lisa Eckey (lisa.eckey@tu-darmstadt.de) is a cryptographer at TU Darmstadt, Germany.

Tyler Moore (tyler-moore@utulsa.edu) is the Tandy Professor of Cyber Security and Information Assurance at The University of Tulsa, OK, USA.

Neha Narula (narula@mit.edu) is Director, Digital Currency Initiative, MIT, Cambridge, MA, USA.

Tim Ruffing (crypto@timruffing.de) is a cryptographer at Blockstream.

Aviv Zohar (avivz@cs.huji.ac.il) is an associate professor of computer science at Hebrew University Jerusalem, Israel.

Copyright held by authors/owners.
Publications rights licensed to ACM.



Watch the authors discuss this work in the exclusive *Communications* video. <https://cacm.acm.org/videos/vulnerability-disclosure>

Bots increasingly tamper with political elections and economic discussions. Tracing trends in detection strategies and key suggestions on how to win the fight.

BY STEFANO CRESCI

A Decade of Social Bot Detection

ON THE MORNING of November 9, 2016, the world woke up to the shocking outcome of the U.S. Presidential election: Donald Trump was the 45th President of the United States of America. An unexpected event that still has tremendous consequences all over the world. Today, we know that a minority of social bots—automated social media accounts mimicking humans—played a central role in spreading divisive messages and disinformation, possibly contributing to Trump’s victory.^{16,19}

In the aftermath of the 2016 U.S. elections, the world started to realize the gravity of widespread deception in social media. Following Trump’s exploit, we witnessed to the emergence of a strident dissonance between the multitude of efforts for detecting and removing bots, and the increasing effects these malicious actors seem to have on our societies.^{27,29} This paradox opens a burning question: *What strategies should we enforce in order to stop this social bot pandemic?*

In these times—during the run-up to the 2020 U.S. elections—the question appears as more crucial than ever. Particularly so, also in light of the recent reported tampering of the electoral debate by thousands of AI-powered accounts.^a

What struck social, political, and economic analysts after 2016—deception and automation—has been a matter of study for computer scientists since at least 2010. In this work, we briefly survey the first decade of research in social bot detection. Via a longitudinal analysis, we discuss the main trends of research in the fight against bots, the major results that were achieved, and the factors that make this never-ending battle so challenging. Capitalizing on lessons learned from our extensive analysis, we suggest possible innovations that could give us the upper hand against deception and manipulation. Studying a decade of endeavors in social bot detection can also inform strategies for detecting and mitigating the effects of other—more recent—forms of online deception, such as strategic information operations and political trolls.

The Social Bot Pandemic

Social bots coexist with humans since the early days of online social networks. Yet, we still lack a precise and

a <https://bit.ly/2BogSgE>

» key insights

- **Social bots are a long studied, yet unsolved, problem in our online social ecosystems and several detection trends appeared through time. The latest and most-promising advance is represented by group-based detectors.**
- **Deception detection is intrinsically adversarial. The application of adversarial machine learning can give us an edge in the fight against all forms of online manipulation and automation.**
- **Recent advances in computing and AI (for example, deepfakes) make individual bots indistinguishable from legitimate users. Future efforts should focus on measuring the extent of inauthentic coordination rather than on trying to classify the nature of individual accounts.**



Facebook's War Room in Menlo Park, CA, on Oct. 17, 2018, ahead of Brazil's runoff election. The company has worked to assuage public concern about the fake accounts, misinformation, and foreign interference that cloud discussion about elections on its site.

well-agreed definition of what a social bot is. This is partly due to the multiple communities studying them and to the multifaceted and dynamic behavior of these entities, resulting in diverse definitions each focusing on different characteristics. Computer scientists and engineers tend to define bots from a technical perspective, focusing on features such as activity levels, complete or partial automation, use of algorithms and AI. The existence of accounts that are simultaneously driven by algorithms and by human intervention led to even more fine-grained definitions and cyborgs were introduced as either bot-assisted humans or human-assisted bots.³ Instead, social scientists are typically more interested in the social or political implications of the use of bots and define them accordingly.

Social bots are actively used for both beneficial and nefarious purposes.¹³

Regarding the detection of benign or malicious social bots, the majority of existing works focused on detecting the latter. The reason is straightforward if we take into account the categorization proposed by Stieglitz et al.³⁰ Bots were categorized according to their *intent* and to their *capacity* of imitating humans, with the majority of existing specimen being either benign bots that do not aim to imitate humans (for example, news and recruitment bots, bots used in emergencies) or malicious ones relentlessly trying to appear as human-operated. The detection of the former category of bots does not represent a challenge, and scholars devoted the majority of efforts to spot the latter, also because of their tampering with our online ecosystems. Indeed, the wide array of actions that social bots perform and the negligible cost for creating and managing them en masse, open up the possibility to de-

ploy armies of bots for information warfare, for artificially inflating the popularity of public characters and for manipulating opinions.

On the onset of the sudden surge of interest around automation and deception, several studies measured the extent of the social bot pandemic. Results are nothing less than worrying. The average presence of bots was estimated to be in the region of 15% of all active Twitter accounts in 2017,³¹ and 11% of all Facebook accounts in 2019³⁸—a considerable share indeed. Even more worrisome, when strong political or economic interests are at stake, the presence of bots dramatically increases. A 2019 study reported that 71% of Twitter users mentioning trending U.S. stocks, are likely to be bots.⁸ Similar results were obtained about the presence of bots in online cryptocurrency discussions²⁴ and as part of the “infodemics” about the COVID-19

Figure 1. The social bot pandemic.

World view of 39 countries for which scientific literature documented political manipulation by social bots. Each country is linked to one or more papers that documented the tampering. Although the list of papers is illustrative and not exhaustive, it nonetheless allows to map the worldwide spread of the social bot pandemic.



Figure 2. Publications per year on the characterization, detection, and impact estimation of social bots.

Since 2014, the number of publications on the topic skyrocketed. We forecast that from 2021 there will be more than one new paper published per day on social bots, which poses a heavy burden on those trying to keep pace with the evolution of this thriving field. Efforts aimed at reviewing and organizing this growing body of work are needed in order to capitalize on previous results.

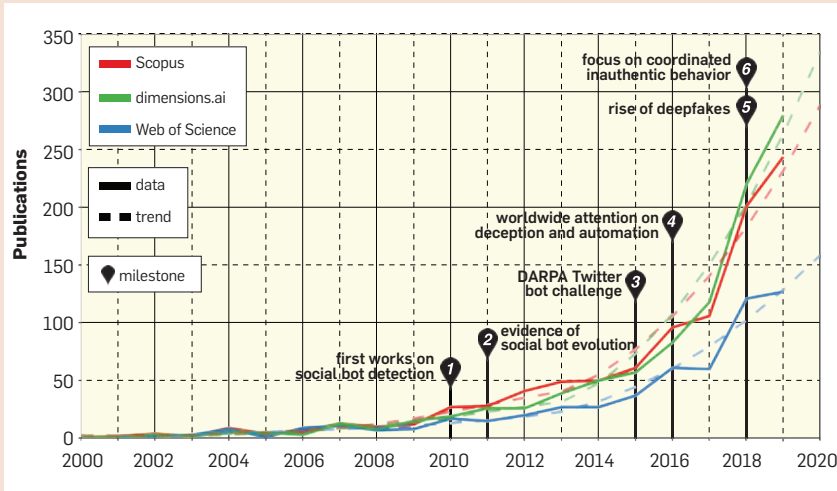
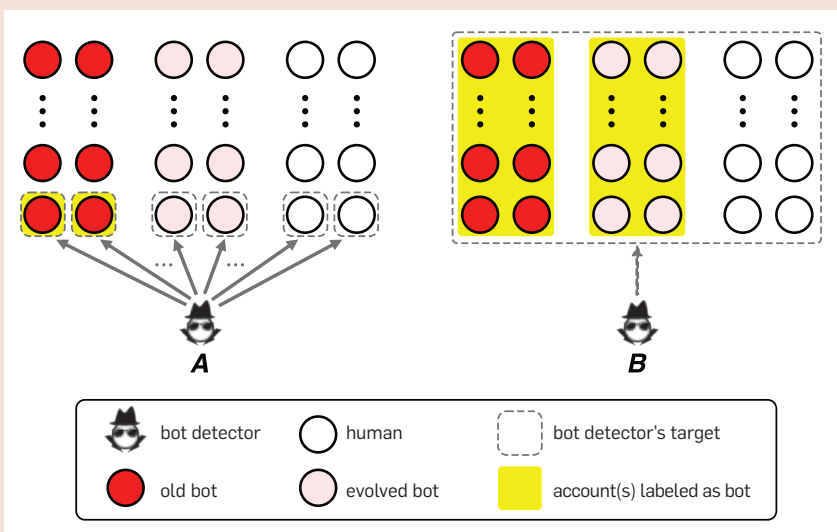


Figure 3. Differences between early and group approaches to social bot detection.

In early approaches (panel A), a supervised detector is separately applied to each account under investigation. If a bot does not appear as markedly different from a human-operated account, as in the case of recent evolved bots, it is likely to evade detection. In more recent approaches (B), a detector analyzes a group of accounts, looking for traces of coordinated and synchronized behaviors. Large groups of coordinated accounts are more likely to be detected than sophisticated individual bots. Nonetheless, prediction errors can still occur for small groups of loosely coordinated bots that might provide insufficient information for detecting them, or for groups of highly coordinated humans that might appear as automated (<https://bit.ly/3gfZucW>). These issues currently represent unsolved challenges in the field.



pandemic.¹⁴ Other studies specifically focused on political activity, concluding that bots played a role in strategic information operations orchestrated ahead of numerous worldwide events, as shown in Figure 1. Despite taking part in political discussions about all countries highlighted in figure, bots did not always have a real impact. In fact, scholars still lack a widespread consensus on the impact of social bots, with some studies reporting on their pivotal role for increasing disinformation’s spread, polarization, and hateful speech,^{27,29} and competing results claiming that bots do not play a significant role in these processes.³² The ubiquity of social bots is also partly fueled by the availability of open source code, for which Bence Kollanyi reported an exponential growth that led in 2016 to more than 4,000 GitHub repositories containing code for deploying Twitter bots.²² Other investigations demonstrated this trend has not halted yet. In fact, by 2018, scholars found more than 40,000 public bot repositories.¹ The looming picture is one where social bots are among the weapons of choice for deceiving and manipulating crowds. These results are backed by the same platforms where information operations took place—namely, Facebook,^b Twitter^c and Reddit^d—that banned tens of thousands accounts involved in coordinated activities since 2016.

Given the reported role of bots in several of the ailments that affect our online ecosystems, many techniques were proposed for their detection and removal—adding to the great coverage from news outlets—contributing to the formation of a steeply rising publication trend. Today, new studies on the characterization, detection, and impact estimation of bots are published at an impressive rate, as shown in Figure 2. Should this skyrocketing trend continue, by 2021 there will be more than one new paper published per day, which poses a heavy burden on those trying to keep pace with the evolution of this thriving field. Perhaps even more importantly, the rate at which new papers are published implies that

b <https://bit.ly/31wtDAK>

c https://about.twitter.com/en_us/values/elections-integrity.html


d <https://bit.ly/38eEgJl>

a huge worldwide effort is taking place in order to stop the spread of the social bot pandemic. *But where is all this effort leading?* To answer this question, we first take a step back at the early days of social bot detection.


The Dawn of Social Bot Detection

The first work that specifically addressed the detection of automated accounts in online social networks dates back to January 2010.³⁷ In the early days, the vast majority of attempts at bot detection featured two distinctive characteristics: they were based on supervised machine learning, and on the analysis of *individual* accounts. In other words, given a group of accounts to analyze, detectors were separately applied to each account of the group, to which they assigned a binary label (either bot or legitimate). This approach to bot detection is schematized in panel A of Figure 3. Here, the key assumption is that bots and humans are clearly separable and that each malicious account has individual features that make it distinguishable from legitimate ones. This approach to the task of social bot detection also revolves around the application of off-the-shelf, general-purpose classification algorithms on the accounts under investigation and on designing effective machine learning features for separating bots from legitimate accounts.

For example, Cresci et al. developed a set of supervised machine learning classifiers for detecting so-called fake followers, a type of automated accounts commonly used to artificially boost the popularity of the public characters that buy them.⁴ Fake followers can be bought for as low as \$12 per 1,000 followers in the surface Web. As a result, they are fairly common.⁶ The authors analyzed some 3,000 fake followers obtained from different vendors and revealed that the simplistic nature of these accounts renders their detection rather easy, even when leveraging only 19 data- and computation-inexpensive features.⁴ After all, fake followers need not perform complex tasks such as producing content or engaging in conversations. Other detection systems make use of large numbers of



Newer bots often feature advanced characteristics that make them much more difficult to detect with respect to older ones.



machine learning features to spot social bots. By leveraging more than 1,200 features of an account, Botometer evaluates possible bots based on their profile characteristics, social network structure, the content they produce, their sentiment expressions, and the timings of their actions.³⁵ Instead of focusing on a specific type of bots, as Cresci et al. did, Botometer represents a “general purpose” bot detector. The generality and ease of deployment of this detector are however counterbalanced by a reduced bot detection accuracy.^{5,17} The two previous detectors simultaneously analyze multiple dimensions of suspicious accounts in order to spot possible bots. Instead, other systems solely focus on network characteristics, textual content of shared messages, or profile information. These systems are typically easier to game, since they only analyze a single facet of the complex behavior of bots.

Despite achieving promising initial results, these early approaches have a number of drawbacks. The first challenge in developing a supervised detector is related to the availability of a ground truth dataset to use in the training phase of the classifier. In most cases, a real ground truth is lacking, and the labels are simply given by human operators that manually analyze the data. Critical issues arise as a consequence of the diverse definitions of social bots, resulting in different labeling schemes.¹⁸ Moreover, humans have been proven to suffer from several annotation biases and to largely fail at spotting recent sophisticated bots, with only 24% bots correctly labeled as such by humans in a recent experiment.⁵ Furthermore, these approaches typically output binary classifications. In many cases however, malicious accounts feature a mixture of automated and human-driven behaviors that cannot be accounted for with simple binary labels. To make matters worse, another major drawback of individual detectors is caused by the *evolutionary* nature of social bots.

The Issue of Bot Evolution

Initial success at social bot detection forced bot developers to put in place sophisticated countermeasures. Because of this, newer bots often feature advanced characteristics that make them much more difficult to detect

e <https://www.nytimes.com/interactive/2018/01/27/technology/social-media-bots.html>

with respect to older ones. This vicious circle leads to the development of always more sophisticated social bots and is commonly referred to as *bot evolution*.

Noteworthy works published by Chao Yang et al. between 2011 and 2013 provided the first evidence and the theoretical foundations to study social bot evolution.³⁴ The first wave of social bots that populated OSNs until around 2011 was made of rather simplistic bots. Accounts with very low reputation due to few social connections and posted messages and featuring clear signs of automation as shown in panel A of Figure 4. Conversely, the social bots studied by Chao Yang et al. appeared as more popular and credible, given the relatively large number of their social connections. In addition, they were no longer spamming the same messages over and over again. Leveraging these findings, authors developed a supervised classifier that was specifically designed for detecting evolving bots. Initially, the classifier proved capable of accurately detecting this second wave of bots. Time went by and new studies acknowledged the rise of a third wave of bots that spread through online social networks from 2016 onward,^{5,13} as shown in panel C of Figure 4. Unfortunately, Yang's classifier for detecting evolving bots was no longer successful at spotting the third wave of malicious accounts.⁶ The previous example serves as anecdotal evidence of bot evolution and of the detrimental effect it has on detectors. Additional quantitative evidence is reported in other studies that evaluated the *survivability* of different bots—that is, their capability of continually evading detection and avoiding being removed from social platforms—and the ability of humans in spotting bots in-the-wild. Results showed that only 5% of newer bots are removed from social platforms, whereas older ones are removed 60% of the times.⁵ Moreover, hundreds of tech-savvy social media users that participated in a crowdsourcing experiment were able to tell apart newer bots from legitimate users only 24% of the times. The same users were instead able of spotting older bots 91% of the times.⁵

The previous anecdotal and quantitative results tell us that current

sophisticated bots are hardly distinguishable from legitimate accounts if analyzed one at a time, as supervised classifiers and crowdsourcing participants did. In fact, *newer bots are more similar to legitimate human-operated accounts than to other older bots*. Among the reasons for the human-like appearance of many bots is an increased hybridization between automated and human-driven behaviors. These cyborgs exist and operate halfway between the traditional concepts of bots and humans, resulting in weakened distinctions and overlapping behaviors between the two. Moreover, they are now using the same technological weapons as their hunters, such as powerful AI techniques for generating credible texts (for example, via the GPT-2 and 3 deep learning models)^f and profile pictures (for example, via StyleGANs deep learning models).^g Indeed, the possibility for malicious accounts to leverage deepfake texts, profile pictures,

f <https://openai.com/blog/better-language-models/>

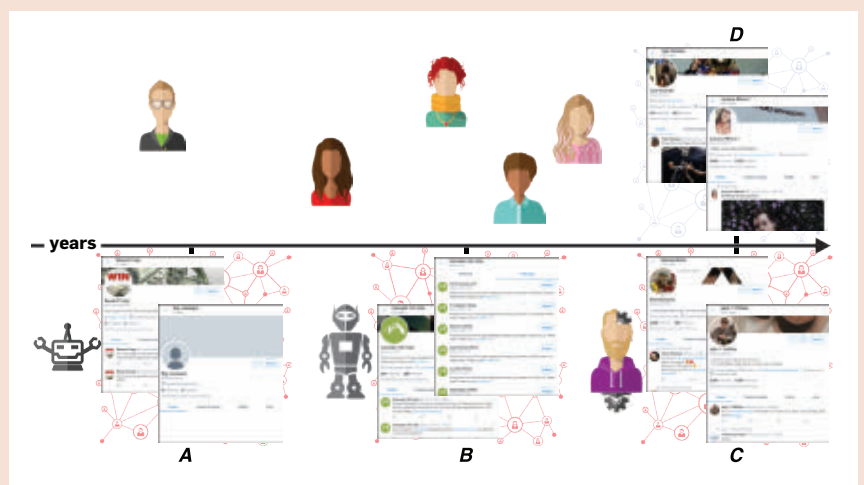
g <https://www.wired.com/story/facebook-removes-accounts-ai-generated-photos/>

and videos is worrying, and worthy of increased attention.¹⁰ Kate Starbird recently discussed a related issue in an inspiring piece on *Nature*.²⁸ Similar to the hazy duality between “bots” and “humans,” she posits that the boundaries between what is “fake” and what is “real,” are blurring. To this end, human-like bots and cyborgs are just the tip of the iceberg, with other newer forms of deception—such as political trolls and “unwitting humans”—that are bound to make the online information landscape an even grimmer place. Figure 4 provides some examples of Twitter profiles that demonstrate how real-world bots evolved over the course of the years. As one form of “social Web virus,” bots mutated thus becoming more resistant to our antibodies. The social bot pandemic gradually became much more difficult to stop. Within this global picture, dichotomous classifications—such as human vs bot, fake vs real, coordinated vs not coordinated—might represent oversimplifications, unable to grasp the complexity of these phenomena and unlikely to yield accurate and actionable results.

Ultimately, the findings about the

Figure 4. Example Twitter profiles showing the issue of bot evolution.

Bots of the first wave (panel A) were very simplistic, with few personal information and social connections. As such, they could be easily distinguished from human-operated legitimate accounts. The second wave consisted of more sophisticated accounts (panel B), featuring detailed personal information. To increase their credibility, these bots often followed one another thus creating clearly identifiable botnets. Nowadays, social bots (panel C) are so carefully engineered as to be more similar to human-operated accounts (panel D) than to other bots. They have large numbers of real friends and followers, they use stolen names and profile pictures, and they intersperse few malicious messages with many neutral ones.



evolution of online automation and deception tell us the naïve assumption of early, supervised bot detection approaches—according to which bots are clearly separable from legitimate accounts—is no longer valid.

The Rise of Group Approaches

The difficulties at detecting sophisticated bots with early approaches rapidly gave rise to a new research trend. Since 2012–2013, several different teams independently proposed new systems that, despite being based on different techniques and implementations, shared the same concepts and philosophy. As schematized in Figure 3 (panel B), the primary characteristic of these new systems, is that of targeting *groups* of accounts as a whole, rather than individual accounts. The rationale for this design choice is that bots act in coordination with other bots, forming botnets to amplify their effects.⁴⁰ The existence of botnets does not necessarily imply that accounts are explicitly connected in the social network, but rather that they are maneuvered by a

single entity and that they share common goals. As such, botnets leave behind more traces of their automation and coordination than those left behind by sophisticated single bots.⁵

Devising techniques for spotting suspiciously coordinated and synchronized behaviors is thus likely to yield better results than analyzing individual accounts. In addition, by analyzing large groups of accounts, detectors also have access to more data for fueling powerful—yet data-hungry—AI algorithms. In 2018, approximately five after the emergence of the group approach to bot detection, also *Facebook*^h and *Twitter*ⁱ acknowledged the importance of focusing on coordinated and inauthentic behaviors. The second common feature to the majority of group detectors is the proposal of important algorithmic contributions, thus shifting from general-purpose machine

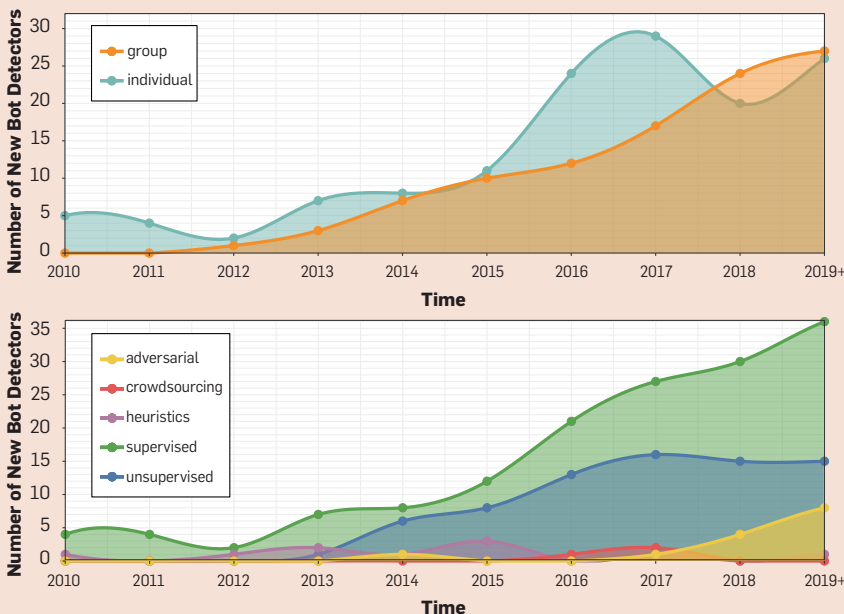
h <https://newsroom.fb.com/news/2018/12/inside-feed-coordinated-inauthentic-behavior/>
 i <https://help.twitter.com/en/rules-and-policies/platform-manipulation>

learning algorithms such as support vector machines and decision trees, to ad-hoc algorithms that are specifically designed for detecting bots, in an effort to boost detection performance. Finally, many group detectors are also based on unsupervised or semi-supervised approaches. Here the idea is to overcome the generalization deficiencies of supervised detectors that are severely limited by the availability of exhaustive and reliable training datasets.¹¹

To quantitatively demonstrate the rise of group approaches to bot detection, Figure 5 illustrates the results of an extensive longitudinal classification. We surveyed more than 230 papers that proposed a bot detection technique and we manually classified each detector along two orthogonal dimensions. The first dimension (panel A) highlights whether detectors target individual accounts or groups of accounts. Then in panel B, we classify detectors according to their high-level approach to the task. In particular, we classified detectors as either based on: heuristics—that is, based on simple rules; crowdsourcing—that is, relying on the judgement of experts; supervised machine learning—such as those based on classification and requiring a labeled training dataset; unsupervised machine learning—such as those based on clustering that do not necessitate of labeled training data; or adversarial approaches—including adversarial machine learning. To better explain our methodology, we provide a couple of examples showing how well-known bot detectors were classified. The system proposed in Ruan et al.²⁶ is designed for detecting compromised accounts—originally legitimate accounts that have been taken over by an attacker. It initially builds a behavioral profile for each investigated account. Then, the system is able to detect compromised accounts via anomaly detection when a behavior diverges significantly with respect to its associated profile. This system is classified as an individual detector (since the behavioral profile of an account depends solely on its own actions) and as an unsupervised detector (since it leverages an anomaly detection technique). Conversely, another system looks for suspiciously

Figure 5. Longitudinal categorization of 236 bot detectors published since 2010.

Data points indicate the number of new detectors per type published in a given year. In panel A, detectors are classified as either focusing on the analysis of individual accounts, or on the analysis of groups of accounts. In panel B, the same detectors are classified based on their high-level approach to the task. Both panels clearly document the rise of a new approach to bot detection, characterized by group-analyses and many unsupervised detectors. Interestingly, the plateau reached by unsupervised approaches since 2017 occurred in conjunction with the recent rise of the adversarial ones.



large similarities between the sequence of activities of vast groups of accounts.⁶ The activity of each account is encoded as a character's string and similarities between account activities are computed by applying the longest common subsequence metric to such strings. Suspiciously long subsequences between activity strings are identified via peak detection, and all those accounts that share the long activity subsequence are labeled as bots. Given these characteristics, this work contributes to group-based bot detectors (since it analyzes a group of accounts looking for similar activity sequences) as well as to unsupervised machine learning approaches (since it leverages an unsupervised peak detection algorithm). Generalizing the two previous examples, we note a few interesting patterns that derive from our classification. The vast majority of techniques that perform network analyses, for instance by considering the social or interactions graph of the accounts, are naturally classified as group based. More often than not, they also propose unsupervised approaches. Contrarily, all techniques based on the analysis of the textual content of posted messages, such as those works that exclusively employ natural language processing techniques, are supervised detectors that analyze individual accounts.

By leveraging classification results reported in Figure 5, we can also derive a number of additional insights. First of all, the rising publication trend of bot detectors follows the general trend of interest around social bots, previously shown in Figure 2. Indeed, since 2015 there has been a steadily increasing number of bot detectors published every year. From the trends shown in panel *A* it is also strikingly evident that group-based approaches, revolving around the analysis of collective behaviors, are increasingly frequent. In fact, in 2018 the number of newly proposed group-based detectors surpassed for the first time that of detectors based on the analysis of individual accounts. From panel *B* we note that bot detection approaches based on heuristics and crowdsourcing received very little attention. This is probably due to the many challenges involved in the development of these systems, which ultimately limit their applicability, scalability and



Devising techniques for spotting suspiciously coordinated and synchronized behaviors is likely to yield better results than analyzing individual accounts.



detection performance. Instead, the number of new supervised detectors has been constantly increasing since 2012, despite their severe generalization issues.¹¹ The adoption of unsupervised machine learning started in 2013 with the rise of group approaches, and now appears to be stationary. Interestingly, the plateau hit by unsupervised approaches co-occurred with the rise of adversarial ones, which might take their place in the coming years. Although the exact number of new bot detectors per type can slightly vary by analyzing a different set of papers, the big picture that emerges from Figure 5—documenting the trends of *individual*, *group* and *adversarial* approaches—is clear, reliable, and insightful.

As a consequence of this paradigm-shift, group-based detectors are particularly effective at identifying evolving, coordinated, and synchronized accounts. For instance, several group detectors implement graph-based approaches and aim at spotting suspicious account connectivity patterns.^{20,24} These techniques are suitable for studying both users interacting with content (for example, retweets to someone else's tweets) or with other users (for example, becoming followers of other users). Coordinated and synchronized behaviors appear as near-fully connected communities in graphs, dense blocks in adjacency matrices, or peculiar patterns in spectral subspaces.²¹ Other techniques adopted unsupervised approaches for spotting anomalous patterns in the temporal tweeting and retweeting behaviors of groups of accounts.^{2,23} One way to spot accounts featuring suspiciously synchronized behaviors is by computing metrics of distance out of the accounts time series, and by subsequently clustering the accounts. The rationale behind this approach is based on evidence suggesting that human behaviors are intrinsically more heterogeneous than automated ones.⁷ Consequently, a large cluster of accounts with highly similar behaviors might indicate the presence of a botnet, even in the absence of explicit connections between the accounts. Distance between accounts time series was computed as a warp-correlation coefficient based on dynamic time warping,² or as the Euclidean distance

between the feature vectors computed by an LSTM autoencoder,²³ a type of deep neural network that is particularly suitable for extracting latent features from sequential data.

As the switch from individual to group detectors demonstrates, the overall approach to the task of bot detection can have serious repercussions on detection performance. At the same time, some scientific communities tend to favor and stick to a specific approach. For instance, works published within the natural language processing community, quite naturally focus on textual content, thus resulting in a multitude of supervised classifiers that analyze accounts individually and that yield binary labels. In contrast, the complex networks community favors graph-based approaches. As a consequence, some combinations of approaches—above all, text-based detectors that perform unsupervised, group analyses—are almost unexplored and definitely underrepresented in the landscape of existing bot detectors. In the future, it would be advisable for multiply efforts to follow the directions that have been mostly overlooked until now.

A Glimpse into the Future of Deception Detection

So far, we highlighted that a shift took place from individual to group detectors in an effort to contrast social bot evolution. Now, we review the latest advances in the field for gaining possible insights into the future of deception detection. We ground this analysis on two observations:

Firstly, we observe that both the individual and the group-based approaches to social bot detection follow a reactive schema. In practice, when scholars and OSN administrators identify a new group of accounts that misbehave and that cannot be effectively detected with existing techniques, they react and begin the development of a new detection system. Hence, the driving factor for the development of new and better detectors have always been bot mischiefs. A major implication of this approach is that improvements in the detection of bad actors typically occur only sometime after having collected evidence of new mischiefs. Bad actors such as bots, cyborgs, and trolls

thus benefit from a long time span—the time needed to design, develop, and deploy a new effective detector—during which they are essentially free to tamper with our online environments. In other words, *scholars and OSN administrators are constantly one step behind of malicious account developers*. This lag between observations and countermeasures possibly explains the current situation with our online social ecosystems: Despite the increasing number of existing detection techniques, the influence of bots and other bad actors on our online discussions did not seem to decrease.

Our second observation is related to the use of machine learning for the task of social bot detection. The vast majority of machine learning algorithms are designed for operating within environments that are stationary and neutral, if not even benign. When the stationarity and neutrality assumptions are violated, algorithms yield unreliable predictions that result in dramatically decreased performances.¹⁵ Notably, *the task of social bot detection is neither stationary nor neutral*. The stationarity assumption is violated by the mechanism of bot evolution that results in accounts exhibiting different behaviors and characteristics over time. Also, the neutrality assumption is clearly violated, since bot developers are actively trying to fool detectors. As a consequence, the very same algorithms that we have been relying upon for a decade, and for which we reported excellent detection results in our studies, are actually seeing their chances to detect bots in-the-wild severely limited.

Recent developments in machine learning may however come to our rescue and may possibly mitigate both issues emerging from the previous observations. Adversarial machine learning is a paradigm specifically devised for application in those scenarios presenting adversaries motivated in fooling learned models.¹⁵ Its high-level goal is to study vulnerabilities of existing systems and possible attacks to exploit them, before such vulnerabilities are effectively exploited by the adversaries. Early detection of vulnerabilities can in turn contribute to the development of more robust detection systems. One practical way to implement this vision is by generating and experimenting with ad-

versarial examples—that is, input instances specifically created to induce errors in machine learning systems.

All tasks related to the detection of online deception, manipulation and automation are intrinsically adversarial. As such, they represent favorable application domains for adversarial machine learning. This intuition resulted in the first papers published in 2018–2019 that initiated the development of an adversarial approach to bot detection, as shown in panel *B* of Figure 5. In the so-called *adversarial bot detection*, scholars experiment with meaningful adversarial examples with which they extensively test the capabilities of current bot detectors.⁹ Within this context, adversarial examples might be sophisticated types of existing bots and trolls that manage to evade detection by current techniques; or even bots that do not exist yet, but whose behaviors and characteristics are simulated, as done by Cresci et al.;⁹ or bots developed ad-hoc for the sake of experimentation, as done by Grimme et al.¹⁷ Finding good adversarial examples can help scholars understand the weaknesses of existing bot detection systems. As a result, bot hunters need not wait anymore for new bot mischiefs in order to adapt their techniques, but instead they can proactively (instead of reactively) test them, in an effort that could quickly make them more robust. In addition, this paradigm accounts for adversaries *by design*, thus providing higher guarantees for deception detection, which violates the stationarity and neutrality assumptions.

The previous analysis highlights that initial efforts toward adversarial bot detection were driven by the creativity of some researchers and only covered few cases with limited applicability.^{9,17} In the near future they could instead be powered by the latest developments in AI. Generative adversarial networks (GANs) are a powerful machine learning framework where two competing deep learning networks are jointly trained in a game-theoretic setting.¹⁵ In particular, a GAN is composed of a *generator* network that creates data instances and a *discriminator* network that classifies data instances, combined as shown in Figure 6 where a GAN is instantiated for a generic task of deception detection. The goal of

the generator is that of creating synthetic data instances that resemble the properties of real organic data, while the typical goal of the discriminator is to classify input data instances as either synthetic or organic. The discriminator is evaluated based on its binary classification performance, while the generator is evaluated in terms of its capacity to induce errors in the discriminator, hence the competition between the two networks.

Originally, GANs were proposed as a form of generative model—that is, the focus was posed on the generator network. A notable example of this kind is represented by the GAN trained in Wu et al.³³ for creating adversarial examples of social bots that improved the training of downstream detectors. However, with the end-goal of providing even larger improvements on deception detection, we could envision the adoption of GANs for training better discriminator networks. In particular, the generator of a GAN could be used as a generative model for creating many plausible adversarial examples, thus overcoming the previously mentioned limitations in this task and the scarcity of labeled datasets. Then, the whole GAN could be used to test the discriminator against the adversarial examples and to improve its detection performances. This paradigm has never been applied to the task of social bot detection, but it was tested with promising results for related tasks, such as that of fake news generation/detection.³⁹ The adversarial framework sketched in Figure 6 is general enough to be applied to a wide set of deception detection tasks, comprising the detection of social bots, cyborgs, trolls and mis/disinformation. Furthermore, in contrast with existing adversarial approaches for bot detection, it is grounded on an established and successful machine learning framework, rather than on ad-hoc solutions lacking broad applicability.

Despite the high hopes placed on adversarial approaches for detecting deception and automation, this research direction is still in its infancy and, probably due to its recency, is still lagging behind more traditional approaches. As such, efforts at adversarial detection can only be successful if the scientific community decides to rise to the many open challenges.

Among them is the development of techniques for creating many different kinds of adversarial examples and to evaluate whether these examples are realistic and representative of future malicious accounts. In spite of these challenges, our analysis and the promising results obtained so far strongly motivate future endeavors in this direction, as also testified by the sparking adversarial trend in Figure 5.

Open Challenges and the Way Ahead

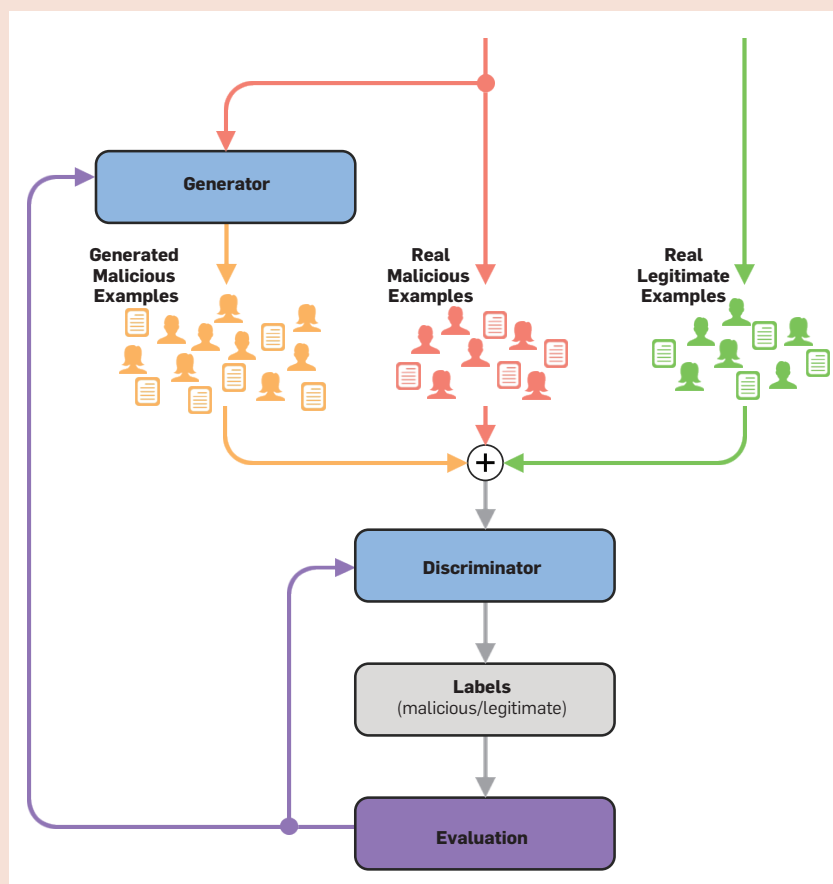
The exponentially growing body of work on social bot detection shown in Figure 2, somehow reassures us that much effort is bound to be devoted in the coming years to the fight of this crucial issue. However, at the same

time it also poses some new challenges. Firstly, it is becoming increasingly important to organize this huge body of work. Doing so would not only contribute to a better exploitation of this knowledge but would also allow researchers to more efficiently provide new solutions by avoiding exploring paths that already proved unsuccessful. To this end, this survey aims to provide a contribution to the critical review and analysis of the vast literature on and beyond this topic.

Secondly, the foreseen increase in publications inevitably implies that more bot detectors will be proposed. With the growing number of disparate detection techniques, it is becoming increasingly important to have standard means, such as benchmarks, frame-

Figure 6. Adversarial deception detection based on generative adversarial networks (GANs).

The generator network is employed for creating a large number of adversarial examples resembling the properties of real malicious examples. The discriminator network is trained to distinguish between malicious (either real or generated) and legitimate examples. By jointly training the two networks, the generator learns to produce more challenging malicious examples while the discriminator improves its overall classification performances since it trains on the challenging examples. This conceptual framework can be applied to many tasks, comprising the detection of disinformation, social bots, and trolls.



works and reference datasets, with which to evaluate and compare them. The present situation is one where we have a suitcase filled with all kinds of tools. Unfortunately, we do not really know how to use them profitably, what the differences are between them, and ultimately, what they are really worth! Buying us yet another tool would not help much. Instead, a few targeted investments aimed at extensively evaluating and comparing our current tools would tremendously increase the usefulness of all our suitcase.

One aspect often overlooked when evaluating bot detectors is their *generalizability*, that is, their capacity of achieving good detection results also for types of bots that have not been originally considered. To this regard, our analysis lays the foundations of a bi-dimensional generalizability space, sketched in Figure 7. A desirable scenario for the near future would involve the possibility to evaluate any new bot detector against many different types of social bots, thus moving along the *y*-axis of Figure 7, following the promising approaches recently developed in Echeverria et al.¹¹ and Yang et al.³⁶ It would also be profitable to evaluate detectors against different versions of current bots, thus somehow simulating

the evolving characteristics of bots. This could be achieved by applying the adversarial approach previously described for creating many adversarial examples, opening up experimentation along the *x*-axis of the generalizability space. Combining these two evaluation dimensions, thus extensively exploring the generalizability space, would allow a much more reliable assessment of the detection capabilities of present and future techniques, thus avoiding overestimates of detection performance. In order to reach this ambitious goal, we must first create reference datasets that comprise several different kinds of malicious accounts, including social bots, cyborgs and political trolls, thus significantly adding to the sparse resources existing as of today.^j Here, challenges include the limited availability of data itself, missing or ambiguous ground truth and the obsolescence of existing datasets that hardly cope with the rapid evolution of malicious accounts. To this regard, continuous data-sharing initiatives such as that of Twitter for accounts involved in information operations,^k are

^j One of the few publicly available bot repositories is hosted at: <https://botometer.iuni.iu.edu/bot-repository/datasets.html>

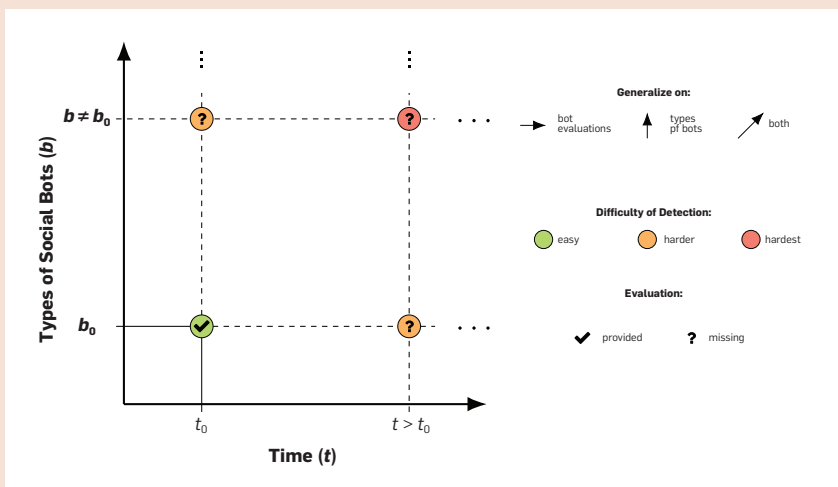
^k <https://transparency.twitter.com/en/information-operations.html>

extremely welcome as they can enable the next wave of research on these issues. Then, we should also devise additional ways for creating a broad array of diverse adversarial examples. Doing so would also require quantitative means to estimate the contributions brought by the different adversarial examples, for instance in terms of their novelty and diversity with respect to existing malicious accounts. These challenges currently stand as largely unsolved and call for the highest effort of our scientific community.

Our longitudinal analysis of the first decade of research in social bot detection revealed some interesting trends. Early days were characterized by simple supervised detectors analyzing accounts *individually*. Unsupervised detectors emerged in 2012–2013 and shifted the target to *groups* of misbehaving accounts. Finally, we highlighted the new rising trend of *adversarial* approaches. Our analysis revealed that for more than a decade we fought each of the menaces posed by sophisticated social bots, cyborgs, trolls, and collusive humans, separately. Now, thanks to the rise of AI-enabled deception techniques such as *deep-fakes*, the most sophisticated of these malicious actors are bound to become indistinguishable from one another, and likely also from legitimate accounts. It is thus becoming increasingly necessary to focus on spotting the techniques used to deceive and to manipulate, rather than trying to classify individual accounts by their nature. Inauthentic coordination is an important piece of the deception puzzle, since it is exploited by bad actors for obtaining visibility and impact. Moreover, it is oblivious to the different types of bad actors. In other words, both our findings and recent reflections^{17,28} suggest that we should keep on moving away from simple supervised approaches focusing on individual accounts and producing binary labels. We should instead take on the challenging task of embracing the complexity of deception, manipulation and automation by devising unsupervised techniques for spotting suspicious coordination. In addition, future techniques should not provide oversimplistic binary labels as often done and just as often

Figure 7. The bi-dimensional generalizability space.

Axes represent dimensions along which to test generalization capabilities of detectors. The majority of existing detectors are evaluated under favorable conditions—that is, only against a specific type of bots (b_0) and with data collected at a specific point in time (t_0)—thus possibly overestimating their capabilities. The actual detection performance for $b \neq b_0$ and for $t > t_0$ are unknown. More realistic estimations could be obtained by evaluating detectors under more general conditions. Generalization along the *y*-axis can be achieved by adopting evaluation methodologies such as that proposed by Echeverria et al.¹¹ Generalization along the *x*-axis can be obtained by applying adversarial approaches aimed at creating variations of currently existing bots.




criticized,¹ but should instead produce multifaceted measures of the extent of suspicious coordination.

Our in-depth analysis revealed the emergence of group-based approaches several years before “coordinated inauthentic behavior” was acknowledged as the main threat to our online social ecosystems by the general public and by the social platforms themselves. Among the most pressing challenges along this line of research is the problem of scalability of group-based detectors and the intrinsic fuzziness of “inauthentic coordination.” In fact, the scalable and generalizable detection of coordination is still a largely open challenge, with only few contributions proposed so far.^{12,25} Similarly, computational means to discriminate between authentic and inauthentic coordination are yet to be proposed and evaluated. Interestingly, the same analysis that anticipated worldwide interest in inauthentic coordination, is now suggesting that adversarial approaches might give us an edge in the long-lasting fight against online deception.

Summarizing the main suggestions stemming from our extensive analysis, future deception detection techniques should: focus on identifying suspicious coordination independently of the nature of individual accounts; avoid providing binary labels in favor of fuzzier and multifaceted indicators; favor unsupervised/semi supervised approaches over supervised ones; and account for adversaries by design. In addition, part of the massive efforts we dedicated to the task of detection should also be reallocated to measure (human) exposure to these phenomena and to quantify the impact they possibly have. Only through enacting these changes we will be able to develop tools that better represent the existing reality, thus providing actionable results to the many scientific communities and stakeholders looking at AI and Big Data tools as a compass to adventure in the perilous landscape of online information. These guiding lights stand in front of us as an exciting and rare opportunity, one that we did not have in the past. Acting upon and capitalizing

on this opportunity is now exclusively on our shoulders.

Acknowledgments. This research is supported in part by the EU H2020 Program under the scheme INFRAIA-01-2018-2019: Research and Innovation action grant agreement #871042 SoBigData++: European Integrated Infrastructure for Social Mining and Big Data Analytics. 

References

1. Assenmacher, D., Clever, L., Frischlich, L., Quandt, T., Trautmann, H. and Grimme, C. *Demystifying Social Bots: On the Intelligence of Automated Social Media Actors*. *Social Media + Society*. SAGE, 2020.
2. Chavoshi, N., Hamooni, H., and Muen, A. DeBot: Twitter bot detection via warped correlation. In *The 16th International Conference on Data Mining* (2016). IEEE, 817–822.
3. Chu, Z., Gianvecchio, S., Wang, H., and Jajodia, S. Detecting automation of Twitter accounts: Are you a human, bot, or cyborg? *IEEE Trans. Dependable and Secure Computing* 9, 6 (2012), 811–824.
4. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. Fame for sale: Efficient detection of fake Twitter followers. *Decision Support Systems* 80 (2015), 56–71.
5. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *Proceedings of the 26th Intern. World Wide Web Conf. Companion* (2017). IW3C2.
6. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. Social fingerprinting: detection of spambot groups through DNA-inspired behavioral modeling. *IEEE Transactions on Dependable and Secure Computing* 15, 4 (2017), 561–576.
7. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. Emergent properties, models, and laws of behavioral similarities within groups of Twitter users. *Computer Communications* 150 (2020), 47–61.
8. Cresci, S., Lillo, F., Regoli, D., Tardelli, S., and Tesconi, M. Cashtag piggybacking: Uncovering spam and bot activity in stock microblogs on Twitter. *ACM Trans. the Web* 13, 2 (2019), 11.
9. Cresci, S., Petrocchi, M., Spognardi, A., and Tognazzi, S. Better Safe Than Sorry: An Adversarial Approach to Improve Social Bot Detection. In *Proceedings of the 11th Intern. Conf. Web Science* (2019). ACM.
10. Da San Martino, G., Cresci, S., Barrón-Cedeño, A., Yu, S., Di Pietro, R., and Nakov, P. 2020. A survey on computational propaganda detection. In *Proceedings of the 29th Intern. Joint Conf. Artificial Intelligence* (2020).
11. Echeverría, J., De Cristofaro, E., Kourtellis, N., Leontiadis, I., Stringhini, G., and Zhou, S. LOBO: Evaluation of generalization deficiencies in Twitter bot classifiers. In *Proceedings of the 34th Annual Computer Security Applications Conf. ACM*, 137–146.
12. Fazil, M. and Abulais, M. A socialbots analysis-driven graph-based approach for identifying coordinated campaigns in Twitter. *J. Intelligent & Fuzzy Systems* 38 (2020), 2961–2977.
13. Ferrara, E., Varol, O., Davis, C., Menczer, F., and Flammini, A. The rise of social bots. *Commun. ACM* 59, 7 (2016), 96–104.
14. Gallotti, R., Valle, F., Castaldo, N., Sacco, P., and De Domenico, M. Assessing the risks of “infodemics” in response to COVID-19 epidemics, 2020; *arXiv:2004.03997* (2020).
15. Goodfellow, I., McDaniel, P., and Papernot, N. Making machine learning robust against adversarial inputs. *Commun. ACM* 61, 7 (July 2018).
16. Gorodnichenko, Y., Pham, T., and Talavera, O. 2018. Social Media, Sentiment and Public Opinions: Evidence from #Brexit and #USElection. Working Paper 24631. National Bureau of Economic Research, 2018.
17. Grimme, C., Assenmacher, D., and Adam, L. Changing perspectives: Is it sufficient to detect social bots? In *Proceedings of the 10th Intern. Conf. Social Computing and Social Media* (2018).
18. Grimme, C., Preuss, M., Adam, L., and Trautmann, H. Social bots: Human-like by means of human control? *Big Data* 5, 4 (2017).
19. Grinberg, N., Joseph, K., Friedland, L., Swire-Thompson, B., and Lazer, D. Fake news on Twitter during the 2016 US presidential election. *Science* 363, 6425 (2019), 374–378.
20. Jiang, M., Cui, P., Beutel, A., Faloutsos, C., and Yang, S. Catching synchronized behaviors in large networks: A graph mining approach. *ACM Trans. Knowledge Discovery from Data* 10, 4 (2016).
21. Jiang, M., Cui, P., Beutel, A., Faloutsos, C., and Yang, S. Inferring lockstep behavior from connectivity pattern in large graphs. *Knowledge and Information Systems* 48, 2 (2016), 399–428.
22. Kollanyi, B. Where do bots come from? An analysis of bot codes shared on GitHub. *Intern. J. Communication* 10 (2016), 20.
23. Mazza, M., Cresci, S., Avenuti, M., Quattrociocchi, W., and Tesconi, M. RTbust: Exploiting temporal patterns for botnet detection on Twitter. In *Proceedings of the 11th Intern. Conf. Web Science* (2019). ACM.
24. Nizzoli, L., Tardelli, S., Avenuti, M., Cresci, S., Tesconi, M., and Ferrara, E. Charting the landscape of online cryptocurrency manipulation. *IEEE Access* 8 (2020), 113230–113245.
25. Pacheco, D., Hui, P., Torres-Lugo, C., Tran Truong, B., Flammini, A., and Menczer, F. 2020. Uncovering coordinated networks on social media, 2020; *arXiv:2001.05658*.
26. Ruan, X., Wu, Z., Wang, H., and Jajodia, S. Profiling online social behaviors for compromised account detection. *IEEE Trans. Information Forensics and Security* 11, 1 (2015), 176–187.
27. Shao, C., Luca Ciampaglia, G., Varol, O., Yang, K., Flammini, A., and Menczer, F. The spread of low-credibility content by social bots. *Nature commun.* 9, 1 (2018), 4787.
28. Starbird, K. Disinformation’s spread: Bots, trolls and all of us. *Nature* 571, 7766 (2019), 449–449.
29. Stella, M., Ferrara, E., and De Domenico, M. Bots increase exposure to negative and inflammatory content in online social systems. In *Proceedings of the National Academy of Sciences* 115, 49 (2018), 12435–12440.
30. Stieglitz, S., Brachten, F., Ross, B., and Jung, A. Do social bots dream of electric sheep? A categorization of social media bot accounts. In *Proceedings of the 17th Australasian Conf. Information Systems* (2017).
31. Varol, O., Ferrara, E., Davis, C., Menczer, F., and Flammini, A. Online human-bot interactions: Detection, estimation, and characterization. In *Proceedings of the 11th Intern. Conf. Web and Social Media*. AAAI, 2017.
32. Vosoughi, S., Roy, D., and Aral, S. The spread of true and false news online. *Science* 359, 6380 (2018), 1146–1151.
33. Wu, B., Liu, L., Yang, Y., Zheng, K., and Wang, X. Using improved conditional generative adversarial networks to detect social bots on Twitter. *IEEE Access* 8 (2020), 36664–36680.
34. Yang, C., Harkreader, R., and Gu, G. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Trans. Information Forensics and Security* 8, 8 (2013), 1280–1293.
35. Yang, K., Varol, O., Davis, C., Ferrara, E., Flammini, A., and Menczer, F. Arming the public with artificial intelligence to counter social bots. *Human Behavior and Emerging Technologies* 1, 1 (2019), 48–61.
36. Yang, K., Varol, O., Hui, P., and Menczer, F. Scalable and generalizable social bot detection through data selection. In *Proceedings of the 34th AAAI Conf. Artificial Intelligence* (2020).
37. Yardi, S. Detecting spam in a Twitter network. *First Monday* 15, 1 (2010).
38. Zago, M., Nespoli, P., Papamartzivanos, D., Gil Perez, M., Gomez Marmol, F., Kambourakis, G., and Martinez Perez, G. Screening out social bots interference: Are there any silver bullets? *IEEE Communications Mag.* 57, 8 (2019), 98–104.
39. Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roessner, F., and Choi, Y. Defending against neural fake news. In *Proceedings of the 33rd Conf. Neural Information Processing Systems* (2019). 9051–9062.
40. Zhang, J., Zhang, R., Zhang, Y., and Yan, G. The rise of social botnets: Attacks and countermeasures. *IEEE Trans. Dependable and Secure Computing* 15, 6 (2016), 1068–1082.

Stefano Cresci (s.cresci@iit.cnr.it) is a researcher at the Institute of Informatics and Telematics of the Italian National Research Council in Pisa, Italy.

Copyright held by author/owner.
Publication rights licensed to ACM.

Publish Your Work Open Access With ACM!

ACM offers a variety of Open Access publishing options to ensure that your work is disseminated to the widest possible readership of computer scientists around the world.



Please visit ACM's website to learn more about ACM's innovative approach to Open Access at:
<https://www.acm.org/openaccess>



Association for
Computing Machinery

research highlights

P. 86

**Technical
Perspective
Analyzing
Smart Contracts
with MadMax**

By Benjamin Livshits

P. 87

MadMax: Analyzing the Out-of-Gas World of Smart Contracts

By Neville Grech, Michael Kong, Anton Jurisevic,
Lexi Brent, Bernhard Scholz, and Yannis Smaragdakis

P. 96

**Technical
Perspective
Two for
the Price of One**

By Paul Beame

P. 97

Lower Bounds for External Memory Integer Sorting via Network Coding

By Alireza Farhadi, Mohammad Taghi Hajiaghayi,
Kasper Green Larsen, and Elaine Shi

Distinguished Speakers Program

A great speaker can make the difference between a good event and a WOW event!

Students and faculty can take advantage of ACM's Distinguished Speakers Program to invite renowned thought leaders in academia, industry and government to deliver compelling and insightful talks on the most important topics in computing and IT today. ACM covers the cost of transportation for the speaker to travel to your event.

speakers.acm.org



Association for
Computing Machinery

Technical Perspective Analyzing Smart Contracts With MadMax

By Benjamin Livshits

SMART CONTRACTS PROVIDE a way to bring computational integrity to executing more or less general-purpose programs. While proposed a long time ago, they have only become popular with the advent of newer blockchain-based systems such as Ethereum with its associated Ethereum Virtual Machine (EVM), and several other similar systems. Smart contracts give the hope of being able to capture complex financial interactions and relationships with the help of executing code. As a result, we have seen a multitude of projects in areas as diverse as law and what is frequently referred to as decentralized finance (DeFi) based on smart contracts.


Somewhat notoriously, smart contracts, because they often directly manage financial transactions, wallets, and transfers, have been subject to vulnerability discovery, with many high-profile vulnerabilities, such as the DAO hack, a highly impactful exploit from mid-2016, where a hacker found a loophole in a smart contract that has led to the theft of about \$70 million. This attack and some of the others have generated a great deal of interest in using static analysis and verification techniques to find bugs and vulnerabilities in contracts before they are allowed to be deployed onto a blockchain (since, after all, contracts are generally immutable as well, making bugs fairly difficult to fix after the fact).

MadMax focuses on a fairly specific aspect of smart contracts, that of *metering*. Metering is an approach to charge for contract execution, which plays the dual role of compensating blockchain participants and of preventing denial-of-service attacks. How to do metering properly is actually quite a hard problem. The EVM proposes a specific way to charge for contact execution, as specified in the

Ethereum yellow paper. Gas is provided for the purpose of contract execution but if not enough gas is provisioned, contract state can be rolled back.

MadMax tackles gas-related vulnerabilities, which permit an attacker to force key contract functionality to run out of gas—effectively performing a permanent denial-of-service attack on the contract. As such, the following paper first effectively discovers a new vulnerability. Second, it proposes a detection approach based on a static analysis (defined with the help of Datalog). MadMax analyses the entirety of smart contracts in the Ethereum blockchain at the time of this writing in just 10 hours and flags vulnerabilities in contracts that hold billions of dollars. The analysis MadMax proposes is fairly precise: manual inspection of a sample of flagged contracts shows that 81% of the sampled warnings do indeed lead to vulnerabilities.

The impact of this work is long-ranging and has some implications for the blockchain industry as a whole. Specifically, the metering approach that is based on gas measurements is a highly imperfect design. Fundamentally, assigning fixed weights to individual instructions is bound to create a mismatch with the specifics of individual hardware architectures.

However, given that blockchain is experiencing rapid adoption, the focus on meeting and out-of-gas attacks of the following paper is well-warranted and more research is needed in this space to both propose new ways to do metering and to fix existing attacks. 

Benjamin Livshits is Chief Scientist of Brave Software and an associate professor at Imperial College London, U.K.

Copyright held by author/owner.

MadMax: Analyzing the Out-of-Gas World of Smart Contracts

By Neville Grech, Michael Kong, Anton Jurisevic, Lexi Brent, Bernhard Scholz, and Yannis Smaragdakis

Abstract

Ethereum is a distributed blockchain platform, serving as an ecosystem for smart contracts: full-fledged intercommunicating programs that capture the transaction logic of an account. A gas limit caps the execution of an Ethereum smart contract: instructions, when executed, consume gas, and the execution proceeds as long as gas is available.

Gas-focused vulnerabilities permit an attacker to force key contract functionality to run out of gas—effectively performing a permanent denial-of-service attack on the contract. Such vulnerabilities are among the hardest for programmers to protect against, as out-of-gas behavior may be uncommon in nonattack scenarios and reasoning about these vulnerabilities is nontrivial.

In this paper, we identify gas-focused vulnerabilities and present MadMax: a static program analysis technique that automatically detects gas-focused vulnerabilities with very high confidence. MadMax combines a smart contract decompiler and semantic queries in Datalog. Our approach captures high-level program modeling concepts (such as “dynamic data structure storage” and “safely resumable loops”) and delivers high precision and scalability. MadMax analyzes the entirety of smart contracts in the Ethereum blockchain in just 10 hours and flags vulnerabilities in contracts with a monetary value in billions of dollars. Manual inspection of a sample of flagged contracts shows that 81% of the sampled warnings do indeed lead to vulnerabilities.

1. INTRODUCTION

Ethereum is a decentralized blockchain platform that can execute arbitrarily-expressive computational *smart contracts*. A smart contract can capture virtually any complex interaction, such as responding to communication from other accounts and dispensing or accepting funds. The possibilities for such programmable logic are endless. It may encode a payoff schedule, investment assumptions, interest policy, conditional trading directives, trade or payment agreements, and complex pricing. Virtually any transactional multiparty interaction is expressible without a need for intermediaries or third-party trust.

Smart contracts typically handle transactions in *Ether*, which is the native cryptocurrency of the Ethereum blockchain with a current market capitalization in tens of billions of dollars. Smart contracts (as opposed to noncomputational “wallets”) hold a considerable portion of the total Ether available in circulation, which makes them ripe targets

for attackers. Hence, developers and auditors have a strong incentive to make extensive use of various tools and programming techniques that minimize the risk of their contract being attacked.

Analysis and verification of smart contracts are, therefore, high-value tasks, possibly more so than in any other application domain. The combination of monetary value and public availability makes the early detection of vulnerabilities a task of paramount importance.

A broad family of contract vulnerabilities concerns *out-of-gas* behavior. Gas is the fuel of computation in Ethereum. Due to the massively replicated execution platform, wasting the resources of others is prevented by charging users for running a contract. Each executed instruction costs gas, which is traded with the Ether cryptocurrency. As a user pays gas upfront, a transaction’s computation may exceed its allotted amount of gas. In that case, the Ethereum Virtual Machine (EVM), which is the runtime environment for compiled smart contracts, raises an out-of-gas exception and aborts the transaction. *A contract is at risk for a gas-focused vulnerability if it has not anticipated (or otherwise does not correctly handle) the possible abortion of a transaction due to out-of-gas conditions.* A vulnerable smart contract may be blocked forever due to the incorrect handling of out-of-gas conditions: re-executing the contract’s function will fail to make progress, re-yielding out-of-gas exceptions, indefinitely. Thus, although an attacker cannot directly appropriate funds, they can cause damage to the contract, locking its balance away in what is, effectively, a denial-of-service attack. Such attacks may benefit an attacker in indirect ways—for example, harming competitors or the ecosystem, amassing fame in a black-hat community, or blackmailing.

In this work, we present MadMax:¹ a static program analysis framework for detecting gas-focused vulnerabilities in smart contracts. MadMax is a static analysis pipeline consisting of a decompiler (from low-level EVM bytecode to a structured intermediate language) and a logic-based analysis specification. MadMax is highly efficient and effective: it analyzes the whole Ethereum blockchain in just 10 hours and reports numerous vulnerable contracts holding a total value exceeding \$2.8B, with high precision, as determined from a random sample.

¹ Available at: <https://github.com/nevillegrech/MadMax>.

The original version of this paper appeared in *Proceedings of the ACM Programming Languages 2 (OOPSLA)* (Nov. 2018).

MadMax is unique in the landscape of smart contract analyzers and verifiers. It is an approach employing cutting-edge *declarative static analysis* techniques (e.g., context-sensitive flow analysis and memory layout modeling for data structures), whereas past analyzers have primarily focused on lightweight static analysis, on symbolic execution, or on full-fledged verification for functional correctness. As MadMax demonstrates, static program analysis offers a unique combination of advantages: very high scalability (applying to the entire blockchain) and high coverage of potential vulnerabilities. Additionally, MadMax is raising the level of abstraction of automated security analysis, by encoding complex properties (such as “safely resumable loop” or “storage whose increase is caused by public calls”), which, in turn, allow detecting vulnerabilities that span multiple transactions.

2. BACKGROUND

A blockchain is a shared, transparent distributed ledger of transactions that is secured using cryptography. One can think of a blockchain as a long and ever-growing list of blocks, each encoding a sequence of individual transactions, always available for inspection and safe from tampering. Each block contains a cryptographic signature of its previous block. Thus, no previous block can be changed or rejected without also rejecting all its successors. Peers/miners run a mining client for separately maintaining the current version of the blockchain. Each of the peers considers the longest valid chain starting from a *genesis* block to be the accepted version of the blockchain. To encourage transaction validation by all peers and discourage wasted or misleading work, a blockchain protocol typically combines two factors: an incentive that is given as a reward to peers successfully performing validation, and a proof-of-work, requiring costly computation to produce a block. To see how distributed consensus and permanent record-keeping arise, consider a malicious client who tries to double-spend a certain amount. The client may propagate conflicting transactions (e.g., paying sellers *A* and *B*) to different parts of the network. As different peers become aware of the two versions of the truth, a majority will arise, because the peers will build further blocks over the version they perceived as current. Thus, a majority will soon accept one of the two spending transactions as authoritative and will reject the other. The minority has to follow suit, or its further participation in growing the blockchain will also be invalidated: the rest of the peers will disregard any of the blocks not resulting in the longest chain.

Using this approach, a blockchain can serve to coordinate all multiparty interactions with trust arising from the majority of peers, instead of being given to an authority by default.

The original blockchain, at least in its popular form, is due to the Bitcoin platform.¹¹ Bitcoin is explicitly a special-purpose cryptocurrency platform. Therefore, the data registered on the Bitcoin ledger can be seen as transaction parties and amounts (with minor logic permitted for cryptographic authentication). In contrast, the blockchain formulation we are interested in is the one popularized by the Ethereum platform^{4, 21}: registered accounts may contain smart contracts,

that is, full-fledged programs that can perform arbitrary computations, enabling the encoding of complex logic.

Ethereum smart contract programming is most commonly done in the Solidity language.¹⁸ Solidity is a JavaScript-like language, enhanced with static types, contracts as a class-like encapsulation construct, contract inheritance, and numerous other features.

The Solidity (or other high-level language) level of abstraction is significantly removed from that of the code that directly runs on the Ethereum blockchain. Instead, Ethereum natively supports a low-level bytecode language—the Ethereum platform is essentially a distributed, replicated virtual machine, called the *Ethereum VM (EVM)*. The EVM is a low-level stack-machine with an instruction set such as standard arithmetic instructions, basic cryptography primitives (mainly cryptographic hashing), primitives for identifying contracts and calling out to different contracts (based on cryptographic signatures), exception-related instructions, and primitives for gas computation. Data is stored either on the blockchain (a memory area called *storage*), in the form of persistent data structures, or in contract-local transient *memory*.

In our work, we focus on analyzing smart contracts at the bytecode level. This is a *high-cost* design decision (due to the low-level nature of the bytecode). At the same time, the EVM bytecode level of abstraction yields a *high payoff* for analyses that target it. A bytecode-level analysis does not require a contract’s source, allowing the analysis of both new and deployed contracts, originally written in any language. At the bytecode level, the input code is normalized, with all control flow being explicit, uniform, and simplified. Furthermore, the impedance mismatch between a high-level language and the EVM bytecode is often a source of confusion and error. For instance, consider the code pattern here:

```
creditorAddresses = new address [] (size);
```

This code RESULTS in iteration over all locations of an array, to set them to zero. This iteration can well run out of gas. (Such code was behind a vulnerability¹ in the GovernMental¹⁶ smart contract, for example.) The iteration is implicit at the Solidity level but immediately apparent at the bytecode level.

3. GAS-FOCUSED VULNERABILITIES

We next identify some of the most common patterns of gas-focused vulnerabilities. We employ Solidity for illustration purposes, even though our entire analysis work is at the EVM bytecode level.

The Ethereum execution model incentivizes users to minimize the number of instructions executed, by making them pay up front for the gas required to execute a transaction. Running out of gas is common, but, in most cases, this is not catastrophic: the transaction is reverted and the end user reruns it with a higher gas budget.

However, Ethereum smart contracts can relatively easily reach a state such that there will never be enough gas to run their code. The most common reason is the block gas limit of the Ethereum network—currently at 9M units of gas, which is enough for a mere few hundred writes to storage (i.e., to the blockchain).

3.1. Unbounded mass operations

The most standard form of a gas-focused vulnerability is that of unbounded mass operations. Loops whose behavior is determined by user input could iterate too many times, exceeding the block gas limit, or becoming too economically expensive to perform. The code may not have predicted this possibility, thus failing to ensure that the contract can continue to operate as desired under these conditions. This will commonly lead to a denial of service for all transactions that must attempt to iterate the loop. Consider the contract:

```
contract NaiveBank {
  struct Account {
    address addr;
    uint balance;
  }
  Account accounts [];

  function applyInterest () returns (uint) {
    for (uint i=0; i<accounts.length; i++) {
      // apply 5 percent interest
      accounts [i].balance =
        accounts [i].balance * 105 / 100;
    }
    return accounts.length;
  }
}
```

As the number of accounts is increased, the gas requirements for executing `applyInterest` will rise. Very quickly (after a mere few hundred entries are added to `accounts`), the function will be impossible to execute without raising an out-of-gas exception: the cost of the loop's instructions exceeds the Ethereum block gas limit.

Ethereum programming safety recommendations¹⁷ suggest that programs should avoid having to perform operations for an unbounded number of clients (instead merely enabling the clients to “pull” from the contract). However, it is easy for contracts to violate this practice, without realizing that a loop's iterations are bounded only by user-controlled quantities.

An alternative recommendation is that when loops do need to perform operations for an unbounded number of clients, the amount of gas should be checked at every iteration and the contract should “keep track of how far [it has] gone, and be able to resume from that point”.¹⁷ This pattern is complex, error-prone, and (as we determine) very uncommon in practice.

3.2. Nonisolated calls (wallet griefing)

An additional way for a contract to run into out-of-gas trouble involves invoking external functionality that may itself throw an out-of-gas exception. The first element of the problem is a call that the programmer may not have considered extensively. Such calls are typically implicit, as part of Ether transfer. Sending Ether involves calling a fallback function on the recipient's side.

It is illustrative to see the issue based on the Solidity primitives and recommended practices. In Solidity, sending Ether is performed via either the `send` or the `transfer` primitive.

These have different ways to handle transfer errors. For instance, `send` returns false if sending Ether fails:

```
<address>.send (uint256) returns (bool)
```

On the other hand, `transfer` raises an error (i.e., throws an exception) if sending Ether fails.

Importantly, both the `send` and the `transfer` Solidity primitives are designed with failure in mind. Both are translated into regular calls at the EVM bytecode level, but with a limited gas budget of 2300 given to the callee. This is barely enough to allow executing some logging code on the recipient's side. Therefore, the emphasis is placed on the error handling.

A good practice locally (and also used in recommended Ethereum security code patterns¹⁷) is using the `send` primitive always with a check of the result and aborting the transaction by throwing an exception, if a `send` fails. This effectively turns a `send` into a `transfer` plus any other code the user wants.

The problem arises when that exception is thrown in the middle of a loop, which is also handling other external accounts. The contract programmer or auditor may easily miss the potential threat. For instance, the loop may iterate only a bounded number of times (e.g., a contest may award money to the three leaders of a scoreboard) tricking the programmer into thinking that its gas consumption is fixed. Furthermore, it is counter-intuitive to consider that an external party will purposely abort the very transaction that gives it money. Finally, the usually-conservative naïve error handling of eagerly aborting the transaction conspires to cause the problem.

We can see the issue in example code for a vulnerability²⁰ appealingly termed *wallet griefing*.² Consider a simple loop that tries to reward the three winners of a contest:

```
for (uint i = 0; i < 3; i++)
  if (!(winners [i].send (reward))) throw;
```

The problem is that the `send` command will also result in the callback function of the winner being executed. All it takes for the contract to be vulnerable is for attackers to make themselves a winner and then provide a callback function that runs out of gas. The sender contract may never be able to recover from such conditions—for example, code clearing the winners may only appear after the end of the above loop.

3.3. Integer overflows

A programming error that commonly expresses itself as a gas-focused vulnerability results from possible integer overflows, often (but not exclusively) arising due to the Solidity-type inference approach. This is a separate pattern from the general attack of Section 3.1, as the iteration is not merely unbounded but literally nonterminating.

² The slang term “griefing” comes from the gaming community, where it is used to denote targeted destructive behavior meant to harass.

Consider the following contract:

```
contract Overflow {
  Payee payees [];

  function goOverAll () {
    for (var i = 0; i < payees.length; i++)
      { ... }
  } ...
}
```

The use of `var` induces a type inference problem. (Newer versions of Solidity statically detect this issue.) The inferred type of variable `i` is `uint8` (i.e., a byte), as the variable is initialized to 0 and `uint8` is the most precise type that can hold 0 while being compatible with all operations on `i`. Unfortunately, this means that a mere addition of 256 members to `payees` is enough to cause the loop to not terminate, quickly resulting in gas exhaustion. An attacker can exploit this vulnerability by adding fake payees using appropriate public functions (not shown) until the overflow is triggered.

4. DECOMPILING EVM BYTECODE

The first step of our gas-focused vulnerability analysis is a decompilation step, raising the level of abstraction from that of EVM bytecode to a structured intermediate language (IR): control-flow graphs (CFGs) over the three-address code. The decompilation step is *itself* a static analysis, as EVM bytecode is low-level: much closer to machine-specific assembly than to structured IRs (e.g., Java bytecode or .NET IL).

4.1. Challenges for EVM bytecode analysis

The EVM is a stack-based low-level IR with minimal structured language characteristics. In the bytecode form of a smart contract, symbolic information has been replaced by numeric constants, functions have been fused together, and control flow is hard to reconstruct. To illustrate, compare the EVM bytecode language to the best-known bytecode language: Java (JVM) bytecode—a much higher-level IR. The design differences include the following:

- Unlike JVM bytecode, EVM does not have structs, classes, or objects, nor does it have a concept of methods.
- Java bytecode is a typed bytecode, whereas EVM bytecode is not.
- In JVM bytecode, the stack depth is fixed under different control flow paths: execution cannot get to the same program point with different stack sizes. In EVM bytecode, no such guarantee exists.
- All control-flow edges (i.e., jumps) in EVM bytecode are to variables, not constants. The destination of a jump is a value that is read from the stack. Therefore, a value-flow analysis is necessary even to determine the connectivity of basic blocks. In contrast, JVM bytecode has a clearly-defined set of targets of every jump, independent of value flow (i.e., independent of stack contents).
- JVM bytecode has defined method invocation and return instructions. In EVM bytecode, although calls to outside a smart contract are identifiable, function calls inside a contract get translated to just jumps (to variable

destinations, per the above point). All functions of a contract are fused in one, with low-level jumps as the means to transfer control.

To call an intracontract function, the code pushes a return address to the stack, pushes arguments, pushes the destination block's identifier (a hash), and performs a jump (which pops the top stack element, to use it as a jump destination). To return, the code pops the caller basic block's identifier from the stack and jumps to it.

4.2. Decompilation approach

MadMax was originally based on the Vandal decompiler.^{3,19} Subsequently, the same analysis logic has been ported to our Gigahorse decompiler framework.⁶

Our decompilation step accepts EVM bytecode as input and produces output in a standard structured intermediate representation: a control-flow graph (of basic blocks and the edges connecting them); three-address code for all operations (instead of operations acting on the stack); and recognized (likely) function boundaries. This representation is encoded as relations (i.e., tables) and queried, recursively, to formulate higher-level program analyses.

We observe that the EVM bytecode input is much like a functional language in continuation-passing-style (CPS) form: all calls and returns are forward calls (jumps), where calls add the continuation (return-to instruction) as one of the arguments. This equivalence of CPS and low-level jumps has been observed before—most explicitly by Thielecke.¹⁵

The technical setting of having CPS input and needing to detect value and control flow is precisely that of *control-flow analysis (CFA)*.^{12,13} Control-flow analysis is also one of the original proposals for a *context-sensitive (call-site sensitive)* static analysis of value flow: for a k -CFA analysis, every call target gets analyzed separately for each caller (i.e., calling instruction), caller's caller, etc., up to a maximum depth, k .

Decompilation, therefore, adopts the standard form of a control-flow analysis,¹³ formulated as an abstract-interpretation. Context sensitivity adapts to the complexity of the input contract, often resulting in analyses with deep context (e.g., $k = 12$). The end result is a three-address code using the schema listed in Figure 1. Syntax sugar and minor detail elision are employed for presentation purposes. Language syntax is quoted using `[` and `]` and implicitly unquoted for meta-variables. For instance, `s:[to:= BINOP(x, y)]` indicates that statement `s` is some binary operation on `x` and `y` with its result in `to`, where `x`, `y`, and `to` are the meta-variables referring to the bytecode variables. The distinction between variables in the analyzed program and meta-variables in the analysis is clear from context; therefore, we simply refer to “variables,” henceforth. We omit the statement identifier, `s`, when it does not affect a rule. We also use `*` as a wildcard, that is, it denotes any variable, which is ignored.

The schema captures all elements of EVM bytecode in a slightly abstracted fashion, using a standard, structured intermediate language. For example, JUMPI instructions have statements, and not arbitrary values, as targets. All binary operations are treated equivalently, as we currently do not attempt to analyze arithmetic expressions. We do not

include unary operations or direct assignment between variables in Figure 1, although we do so in the implementation, because these can be treated as special cases of binary operations. RTVALUE gives a uniform treatment of instructions that return the cost of gas, transaction id, code size, caller, and other run-time quantities.

Figure 1. Domains and decompiler output (i.e., input relations for main analysis).

V is a set of program variables	
C is a set of constants, $C \subseteq \mathbb{Z}$	
S is a set of statement identifiers	
\mathbb{N} is the set of natural numbers, \mathbb{Z} is the set of integers	
<hr/>	
constant assignment	
$s: [to := \text{CONST}(c)]$	$s : S, to : V, c : C$
load from storage	
$s: [to := \text{SLOAD}(index)]$	$s : S, index : V, to : V$
store to storage	
$s: [\text{SSTORE}(from, index)]$	$s : S, index : V, from : V$
load from (volatile) memory	
$s: [to := \text{MLOAD}(index)]$	$s : S, index : V, to : V$
store to (volatile) memory	
$s: [\text{MSTORE}(from, index)]$	$s : S, index : V, from : V$
conditional jump	
$s: [\text{JUMPI}(cond, label)]$	$s : S, cond : V, label : S$
conditional throw	
$s: [\text{THROWI}(cond)]$	$s : S, cond : V$
keccak 256 hash	
$s: [to := \text{SHA3}(ind, len)]$	$s : S, ind : V, len : V, to : V$
call external contract	
$s: [to := \text{CALL}(addr, gas...)]$	$s : S, addr : V, gas : V, to : V$
get remaining gas	
$s: [to := \text{GAS}()]$	$to : V$
get run-time value (e.g. current block size)	
$s: [to := \text{RTVALUE}()]$	$to : V$
CAST integer to a number of bits	
$s: [to := \text{CASTN}(from)]$	$to : V, from : V, n : \mathbb{N}$
binary operator e.g. ϕ, ADD, AND, etc.	
$s: [to := \text{BINOP}(a, b)]$	$y s : S, a : V, b : V, to : V$

5. CORE MADMAX ANALYSIS

The main MadMax analysis operates on the output of decompilation using logic-based specifications. The analysis is implemented in the Datalog language: a logic-based language, equivalent to first-order logic with recursion.⁸ The analysis consists of several layers that progressively infer higher-level concepts about the analyzed smart contract. Starting from the three-address-code representation of Figure 1, concepts such as loops, induction variables, and data flow are first recognized. Then, an analysis of memory and dynamic data structures is performed, inferring concepts such as dynamic data structures, contracts whose storage increases upon reentry, nested arrays, etc. Finally, concepts at the level of analysis for gas-focused vulnerabilities (e.g., loop with unbounded mass storage) are inferred.

5.1. Flow and loop analyses

Ethereum gas-focused vulnerabilities tend to require a high-level semantic understanding of the underlying contract. There are various initial low-level analyses that need to happen before expressing deeper semantics. Thus, the first step of a MadMax analysis is the derivation of loop and data flow information. This yields several relations, on which further analysis steps are built. The relations, together with some extra domain and input context definitions, are given in Figure 2. We do not provide the Datalog rules for any of these relations—their implementation, although not always straightforward, is standard. For instance, it resembles the flow computation in standard Datalog analysis formulations¹⁴ or frameworks for Java bytecode, such as JChord⁹,¹⁰ and Doop.²

The first three computed relations in Figure 2 (INLOOP, INDUCTIONVAR, and LOOPEXITCOND) encode useful concepts in structured loops. Note that loops in low-level programs do not have to be structured; for example, there may not be a loop head that dominates all loop statements. However, Solidity and other EVM languages often produce structured loops as part of their compilation process. The loop analysis finds induction variables, that is, variables that are incremented by a predictable (but not necessarily statically known) amount in each iteration.

The next four relations capture a data-flow analysis. Relation FLOWS expresses a data-flow dependency between variables. In its simplest form, FLOWS is just the reflexive transitive closure of the BINOP input relation; that is, it ignores storage and

Figure 2. Extra domains, input, and output schema for baseline loop and data flow analyses.

F is a set of function hashes	
L is a set of structured loops	
$\text{INPUBLICFUNCTION}(s : S, f : F)$	Statement s is part of function f
$\text{INLOOP}(s : S, l : L)$	Statement s is part of loop l
$\text{INDUCTIONVAR}(v : V, l : L)$	v is an induction variable of loop l
$\text{LOOPEXITCOND}(condVar : V, l : L)$	Loop condition of l is captured by $condVar$
$\text{HASCONSTANTVALUE}(v : V, c : C)$	Constant c may propagate to variable v
$\text{FLOWS}(from : V, to : V)$	Data flow analysis: the value of $from$ flows to to
$\text{VARALIAS}(v : V, u : V)$	Local alias analysis: v, u may be aliased via direct assignment
$\text{MEMCONTENTS}(s : S, p : V, v : V)$	At statement s , contents at memory location p may be v

memory load and store instructions. However, one can give more sophisticated FLOWS definitions without affecting the rest of the analysis. VARALIAS is a similar relation but more restrictive, for variables directly assigned to each other with no further arithmetic. Accordingly, HASCONSTANTVALUE does a simple constant propagation: it is just the composition of VARALIAS with the input CONST relation.

Finally, MEMCONTENTS does a simple analysis of MSTORE operations given the results of VARALIAS and propagates the results to every statement reachable from an MSTORE in the control-flow graph.

There are two points worth mentioning about the above relations:

- The data-flow analysis (i.e., relations HASCONSTANTVALUE, FLOWS, VARALIAS, and MEMCONTENTS) is best-effort, that is, neither sound nor complete. This means that, first, not all possible flows, aliases, etc. are guaranteed to be found: two variables may hold the same value as a result of complex arithmetic, runtime operations, memory load and stores, etc., without the analysis computing this. Second, not all inferences are guaranteed to hold. For example, an inference that is known to hold in one control-flow path but not in another will be optimistically propagated when paths are merged.

The property of being neither sound nor complete carries over to our overall analysis results. MadMax neither guarantees to detect all gas vulnerabilities nor guarantees that every gas vulnerability reported is a real bug. This design choice is well-aligned with the intended purpose of a bug-detecting static analysis—the value of the analysis is not based on its guarantees but on its real-world usefulness.⁵

- Relations FLOWS and VARALIAS are pervasive in the MadMax analysis. Most other relations we shall see henceforth are transitively closed with respect to either FLOWS or (the weaker) VARALIAS. We elide such transitive-closure Datalog rules from our exposition and only focus on the seed logic of each interesting concept.

Armed with the above basic loop and data-flow analyses, we can establish higher-level concepts, such as a loop's bound. This is defined as LOOPBOUNDY in Figure 3. If both an induction variable i and a noninduction variable c flow to a loop exit condition, then we infer that the loop may be bound by the contents of c . A further refinement of this relation is DYNAMICALLYBOUND, which infers which loops are bound by either storage or some other value that is only known at run-time.

Finally, we define predicate POSSIBLYRESUMABLELOOP, to match loops that appear to implement the Ethereum secure coding recommendations,¹⁷ by checking the amount of remaining gas, saving to (permanent) storage an induction variable, and loading the same induction variable from storage. Note that this is not an entirely precise detection of resumable loops—it may well be finding instances of code that just happen to match these abstract conditions, for example, gas check, store, and load of induction variable.

Figure 3. Inferring bound loops and resumable loops.

```

LOOPBOUNDY(loop, var) ←
  INDUCTIONVAR(i, loop),
  !INDUCTIONVAR(var, loop),
  FLOWS(var, condVar),
  FLOWS(i, condVar),
  LOOPEXITCOND(condVar, loop).

DYNAMICALLYBOUND(loop) ←
  [dynVar := SLOAD(*)],
  LOOPBOUNDY(loop, dynVar).

DYNAMICALLYBOUND(loop) ←
  [dynVar := RTVALUE()],
  LOOPBOUNDY(loop, dynVar).

POSSIBLYRESUMABLELOOP(loop) ←
  [gas := GAS()],
  LOOPBOUNDY(loop, gas),
  INDUCTIONVAR(i, loop),
  FLOWS(loaded, i),
  [loaded := SLOAD(*)],
  FLOWS(i, stored),
  [SSTORE(*, stored)].

```

However, the existence of all three conditions is a very strong indication that the programmer has considered the possibility of an out-of-gas exception and has taken precautions to make the loop resumable on a re-execution of the contract function.

5.2. Analysis of memory layout

A faithful modeling of the Ethereum VM memory layout for dynamic data structures is a key part of MadMax. This modeling is necessary for reducing the false-positive rate of the analysis. An intuitive but naïve approach to find gas vulnerabilities may be to flag any contract that contains loops that are “dynamically bound,” or loops where the number of iterations depends on some value stored in storage or passed as external input. However, a precise analysis requires more sophistication. We find experimentally that around half of the currently deployed contracts have dynamically bound loops—but it would be entirely unrealistic to expect that half of smart contracts currently deployed are vulnerable. Instead, for loops that iterate over unbounded data (i.e., data structures), we need to determine whether the data structure could have been populated by an attacker.

The Ethereum virtual machine does not have notions of high-level data structures. Instead, operations on high-level data structures are compiled down to low-level operations on addressable storage. Solidity offers two main kinds of dynamically-sized data structures: dynamically-sized arrays and associative arrays, that is, maps. Although both arrays and maps can be dynamically resized, no mechanism exists for iterating over maps. Therefore, arrays are the primary data structure to model, in order to capture loops that iterate without bounds.

The Ethereum memory layout is highly unconventional from a traditional programming language standpoint, although perfectly reasonable if one considers the specifics of the execution environment (i.e., a segregated, 256-bit

memory space per contract, cryptographic hashing as a primitive). The main idea is that a *key* represents an array. The key is the address of the memory location holding the array's size. At the same time, the key is *hashed* to yield the address of the memory location that holds the array's contents.

Figure 4 depicts an example of storage allocation for a simple contract with two scalar variables and a two-dimensional dynamic array. Fixed-sized data structures in Solidity are stored consecutively in storage as these appear in program order, starting from offset 0. The individual elements in arrays are also stored consecutively in storage; however, the starting offset of the elements requires some calculations to be determined. Due to their unpredictable size, dynamically-sized array types use a KECCAK256 hash function (SHA3) to find the starting position of the array data. The dynamic array value itself occupies an empty slot in storage at some position p . For a dynamic array, this slot stores the number of elements in the array. The array data, however, is located at KECCAK256(p). The implementation of arrays is extended to arbitrarily-nested dynamic data structures, by recursively mapping the above implementation, necessitating a recursive analysis.

MadMax performs an analysis (elided) for modeling the memory layout and identifying dynamic data structures in smart contracts. The outputs of this analysis are shown in Figure 5. Based on these relations, we define key concepts for gas-focused analyses, as shown in Figure 6. An important concept is INCREASEDSTORAGEONPUBLICFUNCTION. Storage variables that are increased and stored in their corresponding storage slot imply that a contract's array size is increased when some public function is invoked. Moreover, we can find loops that iterate over arrays. We define ARRAYITERATOR as a loop that iterates over an array.

5.3. Top level vulnerability queries

The analysis concepts of the previous sections set up the final queries for gas-focused vulnerabilities. These are made precise by combining several distinct concepts. Figure 7 shows the final output relations of the MadMax analysis in slightly simplified (and inlined to single rule) form.

Consider, for instance, the UNBOUNDEDMASSOP logic: it examines whether an array that can grow in size as the result of a public function has contents that are loaded or stored (the FLOWS(*storeOffsetVar*, *index*) allows dereferencing from the beginning of the contents), inside a loop whose bound is based on the array size and that contains an induction variable that affects the address loaded or stored.

The WALLETGRIEFING query is even more precise, requiring a load from the dynamic array, flow of the loaded value to a call whose result is the condition of a throw statement. The call and the throw need to be in the same loop, which also has an induction variable that

affects the address loaded.

Finally, loop overflows are conservatively asserted to be likely if the induction variable is cast to a short integer or ideally one byte. The loop has to be “dynamically bound” to be vulnerable, that is, the number of iterations is determined by some run-time value.

6. IMPACT

Our original MadMax experiments consider all smart contracts available on the Ethereum blockchain on April 9, 2018. We ran MadMax on an idle machine with an Intel Xeon E5-2687W v4 3.00 GHz and 512 GB of RAM. Due to time constraints, we set a cutoff of 20s for decompilation—beyond that time, contracts are considered to time-out.

The contracts flagged for vulnerabilities, combined, contain 7.07 million ETH, or roughly \$2.8 billion.³ In total, there were 6.33 million contract instances deployed at the time of our blockchain scraping, produced from 91.8k unique programs. 4.1% of the contracts are flagged by MadMax as being susceptible to unbounded iteration, 0.12% to wallet griefing, and 1.2% to overflows of loop induction variables.

To estimate a false-positive rate, we manually inspected a subset of the contracts flagged. Our unbiased sampling process involves taking unique bytecode programs and selecting the first and last few contracts by block-hash order. However, a bias factor is introduced by the need to have source code available online—contracts without source code were not considered, as manual inspection of low-level bytecode is highly time-consuming and unreliable.

We select the first 13 contracts, and manual inspection reveals that 11 of these contracts indeed exhibit 13 distinct vulnerabilities, of 16 flagged, for a precision of $13/16 = 81\%$. The exact number is hardly important—a larger sample could have it move a few percentage points up or down. What is important is that the analysis is precise enough to yield a wealth of true vulnerability warnings. By manually inspecting the sampled contracts, we have gained important insights about the effectiveness of MadMax—presented in detail in the MadMax conference publication.⁷

The entire MadMax analysis of the 91.8k contracts took less than 10 hours, running 45 concurrent processes. Subsequent advances of the Gigahorse decompiler have brought this number down by at least a factor of 2. Decompilation currently exhibits time-outs for around 4% of the contracts, depending on the exact settings.

Note that a confirmed vulnerability in a contract does not mean that: (1) exploiting the vulnerabilities is easy or cheap or (2) the vulnerability blocks all Ether in a contract. For instance, the gas required to exploit an unbounded mass

³ The price of ETH/USD and contract balances are both volatile quantities. To fix a reference point, all numbers given are as of April 9th, 2018 (with ETH/USD at \$400.72).

Figure 4. Outputs of data structure analysis.

```
VARINDEXESSTORAGE(S : S, v : V)
ARRAYSIZEVARIABLE(SV : V, arrId : C, kv : V)
ARRAYIDTOSTORAGEINDEX(arrId : C, v : V)
```

Variable v reads or writes to storage at statement s
 Array $arrId$ has its length and address read in sv and kv , respectively
 v holds a storage address that is part of (outermost) array $arrId$

Figure 5. Storage structure and contents (bottom) for given contract (top). sha3 is the keccak256 hash function.

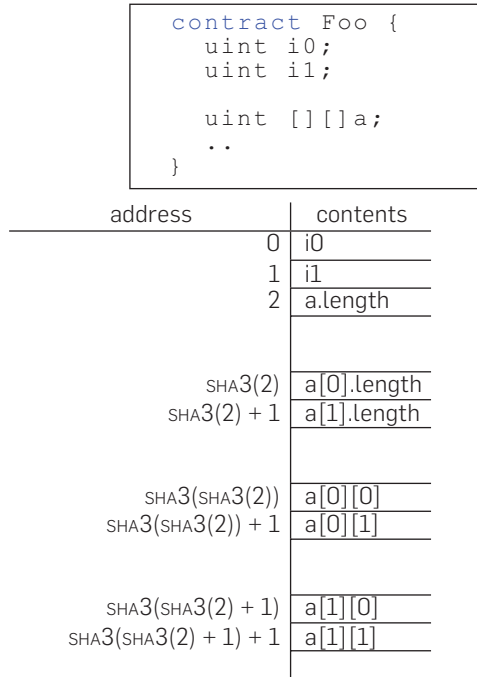


Figure 6. Datalog rules for identifying storage requirements increase in public functions.

```

INCREASEDSTORAGEONPUBLICFUNCTION(arrayId) ←
  ARRAYSIZEVARIABLE(sizeVar, arrayId, keyVar),
  INPUBLICFUNCTION([sizeVar' := ADD(sizeVar, *)], f),
  INPUBLICFUNCTION([SSTORE(keyVar, sizeVar')], f).

ARRAYITERATOR(loop, arrayId) ←
  LOOPBOUNDBy(loop, sizeVar),
  ARRAYSIZEVARIABLE(sizeVar, arrayId, *).
    
```

operation vulnerability may be costly, deterring attackers. However, this does not affect the vulnerable nature of the contract against motivated malicious actors.

7. CONCLUDING DISCUSSION

We presented MadMax, a tool for finding gas-focused vulnerabilities in Ethereum smart contracts. We identify new vulnerabilities for Ethereum smart contracts and demonstrate the first successful design of a static analysis tool at the EVM bytecode level that painstakingly decompiles and reconstructs the program's higher-level semantics. The MadMax approach utilizes best-of-breed techniques and technologies: from abstract-interpretation-based low-level analysis for decompilation to declarative program analysis techniques for higher-level analysis. Our approach is validated using all deployed smart contracts on the blockchain and demonstrates scalability and concrete effectiveness. The threat to some of these smart contracts presented by our tools is overwhelming in financial terms, especially considering the high precision of warnings in a manually-inspected sample.

Gas-focused vulnerabilities are likely to become more

Figure 7. Top-level query for unbounded mass operations, wallet griefing, and overflow vulnerabilities.

```

UNBOUNDEDMASSOP(loop) ←
  INCREASEDSTORAGEONPUBLICFUNCTION(arrayId),
  ARRAYIDTOSTORAGEINDEX(arrayId, storeOffsetVar),
  FLOWS(storeOffsetVar, index),
  VARINDEXESSTORAGE(storeOrLoadStmt, index),
  INLOOP(storeOrLoadStmt, loop),
  ARRAYITERATOR(loop, arrayId),
  INDUCTIONVAR(i, loop),
  FLOWS(i, index),
  !POSSIBLYRESUMABLELOOP(loop).

WALLETGRIEFING(loop) ←
  INCREASEDSTORAGEONPUBLICFUNCTION(arrayId),
  ARRAYIDTOSTORAGEINDEX(arrayId, storeOffsetVar),
  FLOWS(storeOffsetVar, index),
  [loadVar := SLOAD(index)],
  FLOWS(loadVar, target),
  INLOOP([resVar := CALL(target, **)], loop),
  INLOOP([THROWI(condVar)], loop),
  FLOWS(resVar, condVar),
  INDUCTIONVAR(i, loop),
  FLOWS(i, index).

LOOPOVERFLOW(loop) ←
  DYNAMICALLYBOUND(loop),
  [to := CASTN(from, n)], n ≤ 16,
  INDUCTIONVAR(to, loop),
  INDUCTIONVAR(from, loop),
  FLOWS(to, condVar),
  LOOPEXITCOND(condVar, loop).
    
```

relevant in the foreseeable future. Gas (or a quantity like it) is fundamental in blockchain computation and is, for example, included in the design of the upcoming Facebook Libra. Computation under gas constraints requires different coding styles than in traditional programming domains—a simple linear loop over a data structure may render a contract vulnerable! This year, Ethereum's *Istanbul* update makes `SLOAD` four times more expensive, whereas making `SSTORE` cheaper. Exploiting the unbounded operation vulnerability involves many state changing operations to cause the victim to perform more state reading operations. The cost to the attacker is therefore relative to the ratio of the cost of storing against the cost of reading. Hence, this vulnerability will become cheaper to exploit. Moreover, Libra's virtual machine will have state reading operations such as `ImmBorrowField` and `ReadRef`. These will be as expensive as state writing operations `MutBorrowField` and `WriteRef`, which would make the unbounded operations' vulnerability cheaper to exploit in Libra than in Ethereum.

MadMax is the first published analysis to detect threats that require coordination across multiple transactions. This is representative of the future trends for automated security analyses: the analysis will need to account for state changes by independent transactions, long before the final attack can be perpetrated. Furthermore, future threats are likely to involve multicontract or whole-app attacks—for example, with coordination between the off-blockchain part of a decentralized application and its on-blockchain (smart contract) part. This

is a challenging next frontier for security analysis tools. In the case of MadMax, multitransaction reasoning is enabled by positing high-level properties, such as “safely resumable loop.” In turn, this is made possible by the declarative nature of the analysis, which allows a concise, logical specification of complex properties. The same declarative approach may well play an important role in future scaling of analyses to multi-contract, whole-application reasoning.

Acknowledgments

This research was supported partially by the Australian Government through the Australian Research Council’s Discovery Projects funding scheme (project ARC DP180104030). We gratefully acknowledge funding by the European Research Council, grants 307334 and 790340. In addition, the research work disclosed is partially funded by the REACH HIGH Scholars Programme – Post-Doctoral Grants. The grant is part-financed by the European Union, Operational Program II, Cohesion Policy 2014–2020 (Investing in human capital to create more opportunities and promote the well-being of society – European Social Fund). 

References

- Atzei, N., Bartoletti, M., Cimoli, T. A Survey of Attacks on Ethereum Smart Contracts. Technical Report. Cryptology ePrint Archive: Report 2016/1007, <https://eprint.iacr.org/2016/1007>, 2016.
- Bravenboer, M., Smaragdakis, Y. Strictly declarative specification of sophisticated points-to analyses. In *Proceedings of Object Oriented Programming, Systems, Languages, and Applications*, 2009.
- Brent, L., Jurisevic, A., Kong, M., Liu, E., Gauthier, F., Gramoli, V., Holz, R., Scholz, B. Vandal: A scalable security analysis framework for smart contracts. *CoRR*, 2018. abs/1802.08660
- Buterin, V. A next-generation smart contract and decentralized application platform, 2013. <https://github.com/ethereum/wiki/wiki/White-Paper>

- Flanagan, C., Leino, K.R.M., Lillibridge, M., Nelson, G., Saxe, J.B., Stata, R. Extended static checking for Java. In *Proceedings of Programming Language Design and Implementation*. (2002).
- Grech, N., Brent, L., Scholz, B., Smaragdakis, Y. Gigahorse: Thorough, declarative decompilation of smart contracts. In *Proceedings of International Conference on Software Engineering (ICSE)*, 2019.
- Grech, N., Kong, M., Jurisevic, A., Brent, L., Scholz, B., Smaragdakis, Y. Madmax: Surviving out-of-gas conditions in ethereum smart contracts. In *Proceedings of the ACM Programming Languages*, 2 (OOPSLA) (Nov. 2018).
- Immerman, N. Graduate texts in computer science. *Descriptive Complexity*. Springer, 1999.
- Naik, M. Chord: A versatile platform for program analysis. In *Programming Language Design and Implementation*, 2011. Tutorial.
- Naik, M., Park, C., Sen, K., Gay, D. Effective static deadlock detection. In *Proceedings of International Conference on Software Engineering*, 2009.
- Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system, 2009. <https://www.bitcoin.org/bitcoin.pdf>
- Shivers, O. *Control-flow analysis of higher-order languages*. PhD thesis, Carnegie Mellon University (May 1991).
- Shivers, O. Higher-order control-flow analysis in retrospect: lessons learned, lessons abandoned. In *Best of PLDI 1988*. K.S. McKinley, ed. Volume 39, 2004, 257–269
- Smaragdakis, Y., Balatsouras, G. Pointer analysis. *Found. Trends Program. Lang.* 1, 2 (2015), 1–69.
- Thielecke, H. Continuations, functions and jumps. *ACM SIGACT News*, 30 (Jan. 1999), 33–42.
- Various. GovernMental page. <http://governmental.github.io/GovernMental/>
- Various. Safety-ethereum wiki. <https://github.com/ethereum/wiki/wiki/Safety>. Accessed: 2018–04–15.
- Various. GitHub-ethereum/solidity: The solidity contract-oriented programming language, 2018. <https://github.com/ethereum/solidity>
- Various. Vandal—A static analysis framework for ethereum bytecode, 2018. <https://github.com/usyd-blockchain/vandal/>.
- Vessenes, P. Ethereum grieving wallets: Send w/throw is dangerous, 2016. <http://vessenes.com/ethereum-grieving-wallets-send-w-throw-considered-harmful>
- Wood, G. Ethereum: A secure decentralised generalised transaction ledger, 2014. <http://gavwood.com/Paper.pdf>

Neville Grech (me@nevillegrech.com), University of Athens, Greece.

Lexi Brent and Bernhard Scholz ((Lexi.brent, bernhard.scholz@sydney.edu.au)), The University of Sydney, Australia.

Michael Kong and Anton Jurisevic ((mkon1090, ajur4521)@uni.sydney.edu.au), The University of Sydney, Australia.

Yannis Smaragdakis (smaragd@di.uoa.gr), University of Athens, Greece.

Copyright held by authors/owners. Publication rights licensed to ACM.

Semantic Web for the Working Ontologist

Effective Modeling for Linked Data, RDFS, and OWL

**Dean Allemang
James Hendler
Fabien Gandon**

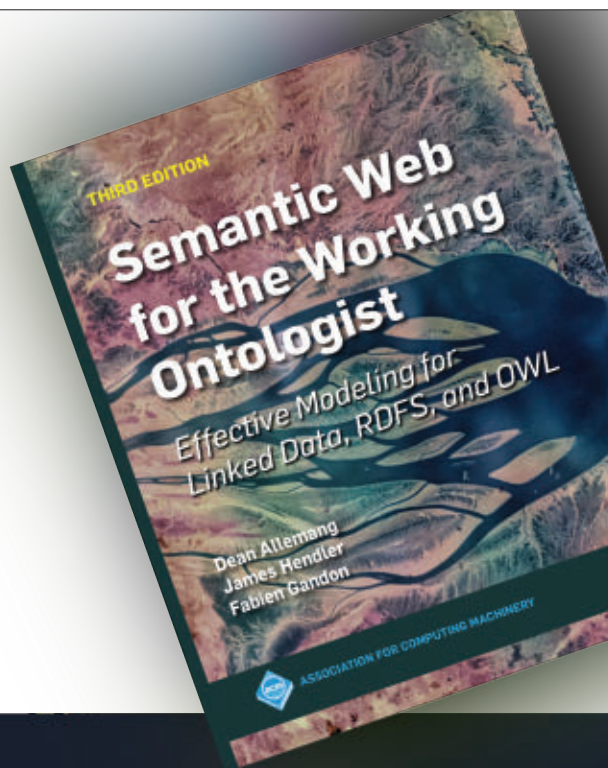
THIRD EDITION

ISBN: 978-1-4503-7617-4

DOI: 10.1145/3382097

<http://books.acm.org>

<http://store.morganclaypool.com/acm>



 **ACM BOOKS**
Collection II

Technical Perspective

Two for the Price of One

By Paul Beame

SORTING AND TRANSMITTING data are two of the most fundamental tasks for which we have employed digital computers. The following paper proves a remarkable connection between how efficiently computers can perform these two tasks, connecting a long-standing question about the optimality of Merge Sort and another, very different, open problem in the study of network coding for data transmission.

Merge Sort was one of the first programs written for digital computers. Though, as a comparison-based in-memory algorithm, it has since been superseded by other sorting algorithms with better memory usage and by algorithms, such as radix sort, that are faster than any comparison-based algorithm, Merge Sort has remained important for sorting large amounts of data that require external storage. It can be modified to merge multiple streams at once and only requires the sequential access common for many storage media. A natural question to ask is: Is (multiway) Merge Sort an optimal choice for an external-memory sorting algorithm, or can we do much better? With the ubiquity of large datasets, this could have many practical applications.

To answer this question, one needs a suitable cost measure. Computers now have many levels of storage hierarchy and hence many levels of “external” memory; data is transferred between levels in blocks rather than individual data items. The cost of those transfers often dominates the cost of operations in “internal” memory. So, a suitable cost measure for external-memory algorithms is the number of transfers of blocks of size B into an internal memory of size M . In 1988, Agarwal and Vitter, who developed the cost measure, showed that $M/2B$ -way Merge Sort, which has transfers that mimic comparisons of an in-memory algorithm for input size n/B , is asymptotically optimal for comparison-based sorting algorithms, even for sorting instances that merely convert a matrix from row-major order to column-major

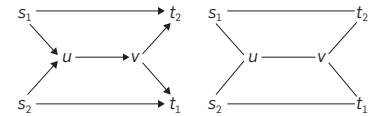
order. The challenge they left is to determine whether this also holds for general external-memory sorting algorithms.

For in-memory algorithms, the gulf between the $O(n \log n)$ time for comparison-based algorithms and that for general algorithms is quite large: Radix sort, which uses indirect addressing, runs in $O(n)$ time when the bit-length w of keys is $O(\log n)$. Also, as the paper notes, other in-memory algorithms that make use of hashing operations on w -bit words can achieve nearly this level of performance for all values of w . It is plausible that a general $O(n)$ time sorting algorithm is achievable for all w (an open question not considered here). However, the operations of indirect addressing and hashing seem to have no analogue for external-memory algorithms, which makes the optimality of Merge Sort plausible.


Though the theory of coding for a single sender and receiver dates back to the earliest days of computing, *network coding* is a more recent invention that arises in the context of many sender-receiver pairs in a shared communication network. Ahlswede et al. showed that, in a directed network, it is possible to send data at a higher rate if nodes in the network actively combine the contents of the data they receive, rather than simply forwarding it as indivisible units. Their classic example is given in

The following paper proves a remarkable connection between how efficiently computers can sort data and how efficiently they can transmit it.

the network on the left below: s_1 and s_2 can simultaneously send streams of messages to t_1 and t_2 respectively if each in-degree 1 node sends its input along both output edges, node u passes on the XOR of its input messages to v , and nodes t_1 and t_2 , in turn, compute the XOR of their input streams.



On the other hand, if the links are undirected, as in the network on the right, one can achieve the same rate without coding (or even using the (u, v) link): each s_i simply uses half the bandwidth of each of the other links to send x_i to t_i . (In this example, the links could be used by alternate message streams in consecutive time steps.) This solution is an example of a (fractional) multi-commodity flow on the network with sender-receiver pairs (s_1, t_1) , (s_2, t_2) , unit demands, and unit capacities. Such a flow reserves a fraction of the capacities on each edge for each sender-receiver pair. The *undirected k -pairs conjecture*, which originated in a 2004 paper of Li and Li, is that such a multi-commodity flow solution is always optimal, so there would never be an advantage to network coding in undirected networks. The surprising result is that if (a weak form of) this network coding conjecture is true, then multiway Merge Sort is asymptotically optimal for external-memory sorting!

Alternatively, it follows from the paper that a better algorithm than Merge Sort would have a second benefit: It could be used to design directed networks for which the rates achievable using network coding are arbitrarily higher than the rates possible without network coding, even with the direction restrictions on the network removed. 

Paul Beame is a professor in the Paul G. Allen School for Computer Science & Engineering at the University of Washington, Seattle, WA, USA.

Copyright held by author.

Lower Bounds for External Memory Integer Sorting via Network Coding

By Alireza Farhadi, Mohammad Taghi Hajiaghayi, Kasper Green Larsen, and Elaine Shi

Abstract

Sorting extremely large datasets is a frequently occurring task in practice. These datasets are usually much larger than the computer's main memory; thus, external memory sorting algorithms, first introduced by Aggarwal and Vitter, are often used. The complexity of comparison-based external memory sorting has been understood for decades by now; however, the situation remains elusive if we assume the keys to be sorted are integers. In internal memory, one can sort a set of n integer keys of $\Theta(\lg n)$ bits each in $O(n)$ time using the classic Radix Sort algorithm; however, in external memory, there are no faster integer sorting algorithms known than the simple comparison-based ones. Whether such algorithms exist has remained a central open problem in external memory algorithms for more than three decades.

In this paper, we present a *tight* conditional lower bound on the complexity of external memory sorting of integers. Our lower bound is based on a famous conjecture in network coding by Li and Li, who conjectured that network coding cannot help anything beyond the standard multicommodity flow rate in undirected graphs.

The only previous work connecting the Li and Li conjecture to lower bounds for algorithms is due to Adler et al. Adler et al. indeed obtain relatively simple lower bounds for *oblivious* algorithms (the memory access pattern is fixed and independent of the input data). Unfortunately, *obliviousness* is a strong limitation, especially for integer sorting: we show that the Li and Li conjecture implies an $\Omega(n \lg n)$ lower bound for internal memory *oblivious* sorting when the keys are $\Theta(\lg n)$ bits. This is in sharp contrast to the classic (nonoblivious) Radix Sort algorithm. Indeed, going beyond obliviousness is highly nontrivial; we need to introduce several new methods and involved techniques, which are of their own interest, to obtain our tight lower bound for external memory integer sorting.

1. INTRODUCTION

Sorting is one of the most basic algorithmic primitives and has attracted lots of attention from the beginning of the computing era. Many classical algorithms have been designed for this problem such as Merge Sort, Bubble Sort, Insertion Sort, etc. As sorting extremely large data has become essential for many applications, there has been a strong focus on designing more efficient algorithms for sorting big datasets² These datasets are often much larger than the computer's main memory and the performance bottleneck changes from being the number of

CPU instructions executed to being the number of accesses to slow secondary storage. In this external memory setting, one usually uses the external memory model to analyze the performance of algorithms. External memory algorithms are designed to minimize the number of *input/output (I/O)s* between the internal memory and external memory (e.g., hard drives and cloud storage), and we measure the complexity of an algorithm in terms of the number of I/Os it performs.

Formally, the external memory model consists of a main memory that can hold M words of w bits each (the memory has a total of $m = Mw$ bits), and an infinite (random access) disk partitioned into blocks of B consecutive words of w bits each (a block has a total of $b = Bw$ bits). The input to an external memory algorithm is initially stored on disk and is assumed to be much larger than M . An algorithm can then read blocks into memory or write blocks to disk. We refer jointly to these two operations as an I/O. The complexity of an algorithm is measured solely in terms of the number of I/Os it makes.

Aggarwal and Vitter² considered the sorting problem in the external memory model. A simple modification to the classic Merge Sort algorithm yields a comparison based sorting algorithm that makes $O((n/B) \lg_{M/B}(n/B))$ I/Os for sorting an array of n comparable records (each storable in a word of w bits). Notice that $O(n/B)$ would correspond to *linear* I/Os, as this is the amount of I/Os needed to read/write the input/output. Aggarwal and Vitter² complemented their upper bound with a matching lower bound, showing that comparison-based external memory sorting algorithms must make $\Omega((n/B) \lg_{M/B}(n/B))$ I/Os. In the same paper, Aggarwal and Vitter also showed that any algorithm treating the keys as *indivisible* atoms, meaning that keys are copied to and from disk blocks, but never reconstructed via bit tricks and the like, must make $\Omega(\min\{n, (n/B) \lg_{M/B}(n/B)\})$ I/Os. This lower bound does not assume a comparison-based algorithm, but instead makes an indivisibility assumption. Notice that the lower bound matches the comparison-based lower bound for large enough B ($B > \lg n$ suffices). The comparison and indivisibility settings have thus been (almost) fully understood for more than three decades.

However, if the input to the sorting problem is assumed to be w bit integers and we allow arbitrary manipulations of the integers (hashing, XOR tricks, etc.), then the situation is completely different. In the standard internal memory computational model, known as the word-RAM, one can design

The original version of this paper is entitled "Lower Bounds for External Memory Integer Sorting via Network Coding" and was published in STOC 2019.

integer sorting algorithms that far outperform comparison-based algorithms regardless of w . More concretely, if the word and key size is $w = \Theta(\lg n)$, then Radix Sort solves the problem in $O(n)$ time, and for arbitrary w , one can design sorting algorithms with a running time of $O(n\sqrt{\lg \lg n})$ in the randomized case⁶ and $O(n \lg \lg n)$ in the deterministic case⁵ (both bounds assume that the word size and key size are within constant factors of each other). In external memory, no integer sorting algorithms faster than the comparison-based $O((n/B) \lg_{M/B}(n/B))$ bound are known! Whether faster integer sorting algorithms exist was posed as an important open problem in the original paper by Aggarwal and Vitter² that introduced the external memory model. Three decades later, we still do not know the answer to this question.

In this paper, we present tight conditional lower bounds for external memory integer sorting via a central conjecture by Li and Li⁷ in the area of *network coding*. Our conditional lower bounds show that it is impossible to design integer sorting algorithms that outperform the optimal comparison-based algorithms, thus settling the complexity of integer sorting under the conjecture by Li and Li.

1.1. Network coding

The field of network coding studies the following communication problem over a network: Given a graph G with capacity constraints on the edges and k data streams, each with a designated source-sink pair of nodes (s_i, t_i) in G , what is the maximum rate at which data can be transmitted concurrently between the source-sink pairs? A simple solution is to forward the data as indivisible packages, effectively reducing the problem to multicommodity flow (MCF). The key question in network coding is whether one can achieve a higher rate by using coding/bit-tricks. This question is known to have a positive answer in directed graphs, where the rate increase may be as high as a factor $\Omega(|G|)$ (by sending XOR's of carefully chosen input bits); see for example, Adler et al.¹ However, the question remains wide open for undirected graphs where there are no known examples for which network coding can do anything better than the multicommodity flow rate. The lack of such examples resulted in the following central conjecture in network coding.⁷

CONJECTURE 1 (UNDIRECTED k -PAIRS CONJECTURE). *The coding rate is equal to the multicommodity flow rate in undirected graphs.*

Despite the centrality of this conjecture, it has so far resisted all attempts at either proving or refuting it. Adler et al.¹ made an exciting connection between the conjecture and lower bounds for algorithms. More concretely, they proved that if Conjecture 1 is true, then one immediately obtains nontrivial lower bounds for all of the following:

- *Oblivious* external memory algorithms
- *Oblivious* word-RAM algorithms
- *Oblivious* two-tape Turing machines

In the above, *oblivious* means that the memory access pattern of the algorithm (or tape moves of the Turing machine) is fixed and independent of the input data. Thus proving

Conjecture 1 would also give the first nontrivial lower bounds for all these classes of algorithms. One can view this connection in two ways: Either as exciting conditional lower bounds for (restricted) algorithms, or as a strong signal that proving Conjecture 1 will be very difficult.

In this paper, we revisit these complexity theoretic implications of Conjecture 1. Our results show that the restriction to oblivious algorithms is unnecessary. In more detail, we show that Conjecture 1 implies nontrivial (and in fact tight) lower bounds for external memory sorting of integers and for external memory matrix transpose algorithms. We also obtain tight lower bounds for word-RAM sorting algorithms when the word size is much larger than the key size, as well as tight lower bounds for transposing a $b \times b$ matrix on a word-RAM with word size b bits. The striking thing is that our lower bounds hold without *any* extra assumptions such as *obliviousness*, *indivisibility*, *comparison-based*, or the like. Thus proving Conjecture 1 is as hard as proving super-linear algorithm lower bounds in the full generality word-RAM model, a barrier far beyond current lower bound techniques! Moreover, we show that the assumption from previous papers about algorithms being oblivious makes a huge difference for integer sorting: We prove an $\Omega(n \lg n)$ lower bound for sorting $\Theta(\lg n)$ bit integers using an oblivious word-RAM algorithm with word size $\Theta(\lg n)$ bits. This is in sharp contrast to the classic (nonoblivious) Radix Sort algorithm, which solves the problem in $O(n)$ time. Thus, the previous restriction to oblivious algorithms may be very severe for some problems.

1.2. Lower bounds for sorting

Our main result for external memory integer sorting is the following connection to Conjecture 1:

THEOREM 2. *Assuming Conjecture 1, any randomized algorithm for the external memory sorting problem with $w = \Omega(\lg n)$ bit integers, having error probability at most $1/3$, must make an expected*

$$\Omega\left(\min\left\{n, \frac{n}{B} \cdot \lg_{2M/B} \frac{n}{B}\right\}\right)$$

I/Os.

Thus if we believe Conjecture 1, then even for randomized algorithms, there is no hope of exploiting integer input to improve over the simple external memory comparison-based algorithms (when $B \geq \lg n$ such that the latter term in the lower bound is the min).

Now observe that because our lower bound only counts I/Os, the lower bound immediately holds for word-RAM algorithms when the word size is some $b = \Omega(\lg n)$ by setting $m = O(b)$ and $B = b/w$ in the above lower bound (the CPU's internal state, i.e., registers, can hold only a constant number of words). Thus, we get the following lower bound:

COROLLARY 3. *Assuming Conjecture 1, any randomized word-RAM algorithm for sorting $w = \Omega(\lg n)$ bit integers, having error probability at most $1/3$ and word size $b \geq w$ bits, must spend*

$$\Omega\left(\min\left\{n, \frac{nw}{b} \cdot \lg \frac{nw}{b}\right\}\right)$$

time.

We note that a standard assumption in the word-RAM is a word size and key size of b , $w = \Theta(\lg n)$ bits. For that choice of parameters, our lower bound degenerates to the trivial $t = \Omega(n)$. This has to be the case, as Radix Sort gives a matching upper bound. Nonetheless, our lower bound shows that when the key size is much smaller than the word size, one cannot sort integers in linear time (recall linear is $O(nw/b)$ as this is the time to read/write the input/output).

Finally, we show that the obliviousness assumption made in the previous paper by Adler et al.¹ allows one to prove very strong sorting lower bounds that even surpass the known (nonoblivious) Radix Sort upper bound:

THEOREM 4. *Assuming Conjecture 1, any oblivious randomized word-RAM algorithm for sorting $\Theta(\lg n)$ bit integers, having error probability at most $1/3$ and word size $\Theta(\lg n)$, must spend $\Omega(n \lg n)$ time.*

Thus, at least for the natural problem of integer sorting, being oblivious has a huge impact on the possible performance of algorithms. Our results are therefore not just an application of the previous technique to a new problem, but a great strengthening. Moreover, as we discuss in Section 3, removing the obliviousness assumption requires new and deep ideas that result in significantly more challenging lower bound proofs.

1.3. Lower bounds for matrix transpose

We also reprove an analog of the lower bounds by Adler et al.¹ for the matrix transpose problem, this time without any assumptions of obliviousness. In the matrix transpose problem, the input is an $n \times n$ matrix A with w -bit integer entries. The matrix is given in row-major order, meaning that each row of A is stored in n/B blocks of B consecutive entries each. The goal is to compute A^T , that is, output the column-major representation of A that stores n/B disk blocks for each column of A , each containing a consecutive range of B entries from the column.

THEOREM 5. *Assuming Conjecture 1, any randomized algorithm for the external memory matrix transpose problem with w bit integer entries, having error probability at most $1/3$, must make an expected*

$$\Omega\left(\min\left\{\frac{n^2 \lg_{2M/B} B}{B}, \frac{n^2 w}{\lg(n^2 w)}\right\}\right)$$

I/Os.

Consider now the matrix transpose problem on the word-RAM with word size b bits (and thus memory size $m = O(b)$). Given an $n \times n$ matrix A with w -bit integer entries, the lower bound in Theorem 5 implies (by setting $B = b/w$):

COROLLARY 6. *Assuming Conjecture 1, any randomized word-RAM algorithm for computing the transpose of an $n \times n$ matrices with w -bit integer entries, having error probability at most $1/3$ and word size b bits, must spend*

$$\Omega\left(\min\left\{\frac{n^2 w \lg(b/w)}{b}, \frac{n^2 w}{\lg(n^2 w)}\right\}\right)$$

time.

2. PRELIMINARIES

We now give a formal definition of the k -pairs communication problem and the multicommodity flow problem.

k -pairs communication problem. To keep the definition as simple as possible, we restrict ourselves to directed acyclic communication networks/graphs and we assume that the demand between every source-sink pair is the same. This will be sufficient for our proofs. For a more general definition, we refer the reader to Adler et al.¹

The input to the k -pairs communication problem is a directed acyclic graph $G = (V, E)$ where each edge $e \in E$ has a capacity $c(e) \in \mathbb{R}^+$. There are k sources $s_1, \dots, s_k \in V$ and k sinks $t_1, \dots, t_k \in V$. Typically, there is also a demand d_i between each source-sink pair, but for simplicity, we assume $d_i = 1$ for all pairs. This is again sufficient for our purposes.

Each source s_i receives a message A_i from a predefined set of messages $A(i)$. It will be convenient to think of this message as arriving on an in-edge. Hence, we add an extra node S_i for each source, which has a single out-edge to s_i . The edge has infinite capacity.

A network coding solution specifies for each edge $e \in E$ an alphabet $\Gamma(e)$ representing the set of possible messages that can be sent along the edge. For a node $v \in V$, define $\text{In}(u)$ as the set of in-edges at u . A network coding solution also specifies, for each edge $e = (u, v) \in E$, a function $f_e: \prod_{e' \in \text{In}(u)} \Gamma(e') \rightarrow \Gamma(e)$ that determines the message to be sent along the edge e as a function of all incoming messages at node u . Finally, a network coding solution specifies for each sink t_i a decoding function $\sigma_i: \prod_{e \in \text{In}(t_i)} \Gamma(e) \rightarrow M(i)$. The network coding solution is correct if, for all inputs $A_1, \dots, A_k \in \prod_i A(i)$, it holds that σ_i applied to the incoming messages at t_i equals A_i , that is, each source must receive the intended message.

In an execution of a network coding solution, each of the extra nodes S_i starts by transmitting the message A_i to s_i along the edge (S_i, s_i) . Then, whenever a node u has received a message a_e along all incoming edges $e = (v, u)$, it evaluates $f_e(\prod_{e \in \text{In}(u)} a_e)$ on all out-edges and forwards the message along the edge e' .

Following Adler et al.¹ (and simplified a bit), we define the *rate* of a network coding solution as follows: Let each source receive a uniform random and independently chosen message A_i from $A(i)$. For each edge e , let A_e denote the random variable giving the message sent on the edge e when executing the network coding solution with the given inputs. The network coding solution achieves rate r if:

- $H(A_i) \geq r d_i = r$ for all i .
- For each edge $e \in E$, we have $H(A_e) \leq c(e)$.

Here $H(\times)$ denotes binary Shannon entropy. The intuition is that the rate is r , if the solution can handle upscaling the entropy of all messages by a factor r compared to the demands.

Multicommodity flow. A multicommodity flow problem in an undirected graph $G = (V, E)$ is specified by a set of k source-sink pairs (s_i, t_i) of nodes in G . We say that s_i is the source of commodity i and t_i is the sink of commodity i . Each edge $e \in E$ has an associated capacity $c(e) \in \mathbb{R}^+$. In addition, there is a demand d_i between every source-sink pair. For simplicity, we assume $d_i = 1$ for all i as this is sufficient for our needs.

A (fractional) solution to the multicommodity flow problem specifies for each pair of nodes (u, v) and commodity i , a flow $f_i(u, v) \in [0, 1]$. Intuitively, $f_i(u, v)$ specifies how much of commodity i is to be sent from u to v . The flow satisfies *flow conservation*, meaning that:

- For all nodes u that is not a source or sink, we have $\sum_{w \in V} f_i(u, w) - \sum_{w \in V} f_i(w, u) = 0$.
- For all sources s_i , we have $\sum_{w \in V} f_i(s_i, w) - \sum_{w \in V} f_i(w, s_i) = 1$.
- For all sinks, we have $\sum_{w \in V} f_i(w, t_i) - \sum_{w \in V} f_i(t_i, w) = 1$.

The flow also satisfies that for any pair of nodes (u, v) and commodity i , there is only flow in one direction, that is, either $f_i(u, v) = 0$ or $f_i(v, u) = 0$. Furthermore, if (u, v) is not an edge in E , then $f_i(u, v) = f_i(v, u) = 0$. A solution to the multicommodity flow problem achieves a rate of r if:

- For all edges $e = (u, v) \in E$, we have $r \cdot \sum_i d_i(f_i(u, v) + f_i(v, u)) = r \cdot \sum_i (f_i(u, v) + f_i(v, u)) \leq c(e)$.

Intuitively, the rate is r if we can upscale the demands by a factor r without violating the capacity constraints.

The undirected k -pairs conjecture. Conjecture 1 implies the following for our setting: Given an input to the k -pairs communication problem, specified by a directed acyclic graph G with edge capacities and a set of k source-sink pairs with a demand of 1 for every pair, let r be the best achievable network coding rate for G . Similarly, let G' denote the undirected graph resulting from making each directed edge in G undirected (and keeping the capacities, source-sink pairs and a demand of 1 between every pair). Let r' be the best achievable flow rate in G' . Conjecture 1 implies that $r \leq r'$.

Having defined coding rate and flow rate formally, we also mention that the result of Braverman et al.⁴ implies that if there exists a graph G where the network coding rate r and the flow rate r' in the corresponding undirected graph G' satisfy $r \geq (1 + \epsilon)r'$ for a constant $\epsilon > 0$, then there exists an infinite family of graphs $\{G^*\}$ for which the corresponding gap is at least $(\lg |G^*|)^c$ for a constant $c > 0$. So far, all evidence suggests that no such gap exists, as formalized in Conjecture 1.

3. PROOF OVERVIEW

In this section, we give an overview of the main ideas in our proof and explain the barriers we overcome in order to remove the assumption of obliviousness. To prove our lower bound for external memory sorting, we focus on the easier problem of permuting. In the permutation problem, we are given an array A of n entries. The i 'th entry of A stores a w -bit data item d_i and a destination $\pi(i)$. The destinations $\pi(i)$ form a permutation π of $\{1, \dots, n\}$. The goal is to produce the output array C where d_i is stored in entry $C[\pi(i)]$. The arrays A and C are both stored in disk blocks, such that each disk block of A stores $b/(\lg n + w)$ entries, and each disk block of C stores b/w entries (the maximum number of entries that can be packed in a block). A sorting algorithm that can sort $(\lg n + w)$ bit integer keys can be used to solve the permutation problem by replacing each entry $(\pi(i), d_i)$ with the integer $\pi(i) \times 2^w + d_i$ (in the addition, we think of d_i as an integer in $[2^w]$). Thus, it suffices to prove lower bounds for permuting.

Consider now an algorithm \mathcal{A} for permuting, and assume for simplicity that it is deterministic and always correct. As in the previous work by Adler et al.¹, we define a graph $G(\mathcal{A})$ that captures the memory accesses of \mathcal{A} on an input array A . The graph G has a node for every block in the input array, a node for every block in the output, and a node for every intermediate block written/read by \mathcal{A} . We call these block nodes. Moreover, the graph has a memory node that represents the memory state of \mathcal{A} . The idea is that whenever \mathcal{A} reads a block into memory, then we add a directed edge from the corresponding block node to the memory node. When \mathcal{A} writes to a block, we create a new node (that replaces the previous version of the block) and add a directed edge from the memory node to the new node. The algorithm \mathcal{A} can now be used to send messages between input and output block nodes as follows: Given messages X_1, \dots, X_n of w bits each and an intended output block node (storing $C[\pi(i)]$) for each message i , we can transmit the message X_i from the input block node representing the array entry $A[i]$ to the output block node representing the array entry $C[\pi(i)]$ simply by simulating the algorithm \mathcal{A} : Each block node of the network always forwards any incoming message to the memory node along its outgoing edge. The memory node thus receives the contents of all blocks that it ever reads. It can therefore simulate \mathcal{A} . Whenever it performs a write operation, it sends the contents along the edge to the designated block node. By the correctness of \mathcal{A} , this results in every output block node knowing the contents of all array entries $C[\pi(i)]$ that should be stored in that output block. Examining this simulation, we see that we need a capacity of b bits on all edges for the simulation to satisfy capacity constraints. Moreover, by the definition of network coding rate (Section 2), we see that the coding rate is w bits.

The idea is that we want to use Conjecture 1 to argue that the graph G must be large (i.e., there must be many I/Os). To do so, we would like to argue that if we undirect G , then there is a permutation π such that for many pairs $A[i]$ and $C[\pi(i)]$, there are no short paths between the block nodes storing $A[i]$ and $C[\pi(i)]$. If we could argue that for $n/2$ pairs $(A[i], C[\pi(i)])$, there must be a distance of at least ℓ steps in the undirected version of G , then to achieve a flow rate of w , it must be the case that the sum of capacities in G is at least $\ell wn/2$. But each I/O adds only $2b$ bits of capacity to G . Thus, if \mathcal{A} makes t I/Os, then it must be the case that $tb = \Omega(\ell wn) \Rightarrow t = \Omega((nw/b) \times \ell) = \Omega((n/B) \times \ell)$.

Unfortunately, we cannot argue that there must be a long path between many pairs in the graph G we defined above. The problem is that the memory node is connected to all block nodes and thus the distance is never more than 2. To fix this, we change the definition of G slightly: After every m/b I/Os, we deactivate the memory node and create a new memory node to replace it. Further I/Os insert edges to and from this new memory node. In order for the new memory node to continue the simulation of \mathcal{A} , the new memory node needs to know the memory state of \mathcal{A} . Hence, we insert a directed edge from the old deactivated memory node to the new memory node. The edge has capacity m bits. Thus, in the simulation, when the current memory node has performed m/b I/Os, it forwards the

memory state of \mathcal{A} to the next memory node who continues the simulation. The m/b I/Os between the creation of new memory nodes has been chosen such that the amortized increase in capacity due to an I/O remains $O(b)$.

We have now obtained a graph G where the degrees of all nodes are bounded by $2m/b$. Thus, for every node G , there are at most $(2m/b)^\ell$ nodes within a distance of ℓ . Thus, intuitively, a random permutation π should have the property that for most pairs $(A[i], C[\pi(i)])$, there will be a distance of $\ell = \Omega(\lg_{2m/b} n/B)$ between the corresponding block nodes. This gives the desired lower bound of $t = \Omega((n/B) \times \ell) = \Omega((n/B) \times \lg_{2m/b} n/B)$.

If we had assumed that the algorithm \mathcal{A} was oblivious as in previous work, we would actually be done by now. This is because, under the obliviousness assumption, the graph G will be the same for all input arrays. Thus, one can indeed find the desired permutation π where there is a large distance between most pairs $(A[i], C[\pi(i)])$. Moreover, all inputs corresponding to that permutation π and data bit strings d_1, \dots, d_n can be simulated correctly using \mathcal{A} and the graph G . Hence, one immediately obtains a network coding solution. However, when \mathcal{A} is not constrained to be oblivious, there can be a large number of distinct graphs G resulting from the execution of \mathcal{A} .

To overcome this barrier, we first argue that even though there can be many distinct graphs, the number of such graphs is still bounded by roughly $(nw/b + t)^\ell$ (each I/O chooses a block to either read or write and there are t I/Os). This means that for $t = o(n)$, one can still find a graph G that is the result of running \mathcal{A} on many different input arrays A . We can then argue that among all those inputs A , there are many that all correspond to the same permutation π and that permutation π has the property from before that, and for most pairs $(A[i], C[\pi(i)])$, there will be a distance of $\ell = \Omega(\lg_{2m/b} n/B)$ between the corresponding block nodes. Thus, we would like to fix such a permutation and use \mathcal{A} to obtain a network coding solution. The problem is that we can only argue that there are many data bit strings d_1, \dots, d_n that together with π result in an array A for which \mathcal{A} uses the graph G . Thus, we can only correctly transmit a large collection of messages, not all messages. Let us call this collection $\mathcal{F} \subseteq \{\{0, 1\}^w\}^n$ and let us assume $|\mathcal{F}| \geq 2^{nw-o(nw)}$. Intuitively, if we draw a uniform random input from \mathcal{F} , then we should have a network coding solution with a rate of $w - o(w)$. The problem is that the definition of network coding requires the inputs to the nodes to be independent. Thus, we cannot immediately say that we have a network coding solution with rate $w - o(w)$ by solving a uniform random input from \mathcal{F} . To remedy this, we instead take the following approach: We let each data bit string d_i be a uniform random and independently chosen w -bit string. Thus, if we can solve the network coding problem with these inputs, then we indeed have a network coding solution. We would now like to find an efficient way of translating the bit strings d_1, \dots, d_n to new bit strings d'_1, \dots, d'_n with $d'_1, \dots, d'_n \in \mathcal{F}$. The translation should be such that each input block node can locally compute the d'_i , and the output block nodes should be able to revert the transformation, that is, compute from d'_i the original bit string

d_i . To achieve this, we need to modify G a bit. Our idea is to introduce a coordinator node that can send short descriptions of the mappings between the d_i s and d'_i s. We accomplish this via the following lemma:

LEMMA 7. *Consider a communication game with a coordinator u , a set $\mathcal{F} \subseteq \{0, 1\}^{nw}$ and n players. Assume $|\mathcal{F}| \geq 2^{nw-r}$ for some r . The coordinator receives as input n uniform random bit strings X_i of w bits each, chosen independently of the other X_j . The coordinator then sends a prefix-free message R_i to the i 'th player for each i . From the message R_i alone (i.e., without knowing X_i), the i 'th player can then compute a vector $\tau_i \in \{0, 1\}^w$ with the property that the concatenation $q := (\tau_1 \oplus X_1) \square (\tau_2 \oplus X_2) \circ \dots \circ (\tau_n \oplus X_n)$ satisfies $q \in \mathcal{F}$, where \oplus denotes bit wise XOR. There exists such a protocol where*

$$\sum_i \mathbb{E}[|R_i|] = O(n + \sqrt{nr} \lg(nw/r)).$$

In particular, if $r = o(nw)$ and $w = \omega(1)$, then the communication satisfies $\sum_i \mathbb{E}[|R_i|] = o(nw)$.

We use the lemma as follows: We create a coordinator node u that is connected to all input block nodes and all output block nodes. In a simulation of \mathcal{A} , the input block nodes start by transmitting their inputs to the coordinator node u . The coordinator then computes the messages in the lemma and sends R_i back to the input block node storing $A[i]$ as well as to the output block node storing the array entry $C[\pi(i)]$. The input block nodes can now compute $d'_i = \tau_i \oplus d_i$ to obtain an input $d'_1, \dots, d'_n \in \mathcal{F}$. We can then run algorithm \mathcal{A} because this is an input that actually results in the graph G . Finally, the output block nodes can revert the mapping by computing $d_i = \tau_i \oplus d'_i$. Thus, what the lemma achieves is an efficient way of locally modifying the inputs of the nodes, so as to obtain an input for which the algorithm \mathcal{A} works. We find this contribution very novel and suspect it might have applications in other lower bound proofs.

The introduction of the node u of course allows some flow to traverse paths not in the original graph G . Thus, we have to be careful with how we set the capacities on the edges to and from u . We notice that edges from the input nodes to u need only a capacity of w bits per array entry (they send the inputs), and edges out of u need $\mathbb{E}[|R_i|]$ capacity for an input d_i (one such edge to the input block node for array entry $A[i]$ and one such edge to the output block node for array entry $C[\pi(i)]$). The crucial observation is that any flow using the node u as an intermediate node must traverse at least two edges incident to u . Hence, only $(nw + 2 \sum_i \mathbb{E}[|R_i|]) / 2$ flow can traverse such paths. If $|\mathcal{F}| \geq 2^{nw-o(nw)}$, then Lemma 7 says that this is no more than $nw/2 + o(nw)$ flow. There therefore remains $nw/2 - o(nw)$ flow that has to traverse the original length $\ell = \Omega(\lg_{2m/b} n/B)$ paths and the lower bound follows.

One may observe that our proof uses the fact that the network coding rate is at most the flow rate in a strong sense. Indeed, the introduction of the node u allows a constant fraction of the flow to potentially use a constant length path. Thus, it is crucial that the network coding rate r and flow rate r' is conjectured to satisfy $r \leq r'$ and not, for example, $r \leq 3r'$.

Indeed, we can only argue that a too-good-to-be-true permutation algorithm yields a graph in which $r \geq ar'$ for some constant $a > 1$. However, Braverman et al.⁴ recently proved that if there is a graph where $r \geq (1 + \varepsilon)r'$ for a constant $\varepsilon > 0$, then there is an infinite family of graphs $\{G'\}$ where the gap is $\Omega((\lg |G'|)^c)$ for a constant $c > 0$. Thus, a too-good-to-be-true permutation algorithm will indeed give a strong counter example to Conjecture 1.

Our proof of Lemma 7 is highly nontrivial and is based on the elegant proof of the \sqrt{IC} bound by Barak et al.³ for compressing interactive communication under nonproduct distributions. Our main idea is to argue that for a uniform random bit string in $\{0, 1\}^{nw}$ (corresponding to the concatenation $X = X_1 \circ \dots \circ X_n$ of the X_i 's in the lemma), it must be the case that the expected Hamming distance to the nearest bit string Y in \mathcal{F} is $O(\sqrt{nmwr})$. The coordinator thus finds Y and transmits the XOR $X \oplus Y$ to the players. The XOR is sparse and thus the message can be made short by specifying only the nonzero entries. Proving that the expected distance to the nearest vector is $O(\sqrt{nmwr})$ is the main technical difficulty and is the part that uses ideas from protocol compression.

4. EXTERNAL MEMORY LOWER BOUNDS

As mentioned in the proof overview in Section 3, we prove our lower bound for external memory sorting via a lower bound for the easier problem of permuting: An input to the permutation problem is specified by a permutation π of $\{1, 2, \dots, n\}$ as well as n bit strings $d_1, \dots, d_n \in \{0, 1\}^w$. We assume $w \geq \lg n$ such that all bit strings may be distinct. The input is given in the form of an array A where the i 'th entry $A[i]$ stores the tuple $(\pi(i), d_i)$. We assume the input is given in the following natural way: Each $\pi(i)$ is encoded as a $\lceil \lg n \rceil$ -bit integer and the d_i 's are given as they are—using w bits for each.

The array A is presented to an external memory algorithm as a sequence of blocks, where each block contains $\lfloor b/(w + \lg n) \rfloor$ consecutive entries of A (the blocks have $b = Bw$ bits). For simplicity, we henceforth assume $(w + \lg n)$ divides b .

The algorithm is also given an initially empty output array C . The array C is represented as a sequence of n words of w bits each, and these are packed into blocks containing b/w words each. The goal is to store $d_{\pi^{-1}(i)}$ in $C[i]$. That is, the goal is to copy the bit string d_i from $A[i]$ to $C[\pi(i)]$. We say that an algorithm \mathcal{A} has an error of ε for the permutation problem, if for every input to the problem, it produces the correct output with the probability at least $1 - \varepsilon$.

The best known upper bounds for the permutation problem work also under the *indivisibility* assumption. These algorithms solve the permutation problem in

$$\begin{aligned} & O\left(\min\left\{n, \frac{nw}{b} \cdot \lg_{m/b}(nw/b)\right\}\right) \\ & = O\left(\min\left\{n, \frac{n}{B} \cdot \lg_{M/B}(n/B)\right\}\right) \end{aligned}$$

I/Os.² Moreover, this can easily be shown to be optimal under the indivisibility assumption by using a counting

argument.² The n bound is the bound obtained by running the naive “internal memory” algorithm that simply puts each element into its correct position one at a time. The other term is equivalent to the optimal comparison-based sorting bound (one thinks of d_i as an integer in $[2^w]$ and concatenates $\pi(i) \circ d_i = \pi(i) \times 2^w + d_i$ and sorts the sequence). Thus, any sorting algorithm that handles $(\lg n + w)$ -bit keys immediately yields a permutation algorithm with the same number of I/Os. We thus prove lower bounds for the permutation problem and immediately obtain the sorting lower bounds as corollaries.

We thus set out to use Conjecture 1 to provide a lower bound for the permutation problem in the external memory model. Throughout the proof, we assume that $nw/b = n/B$ is at least some large constant. This is safe to assume, as otherwise we only claim a trivial lower bound of $\Omega(1)$.

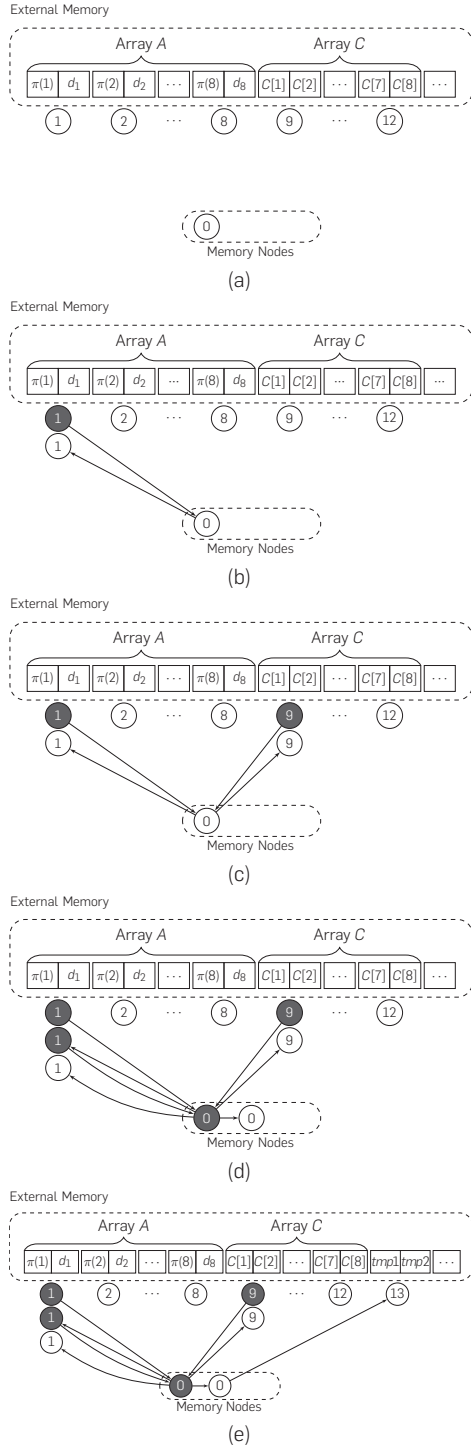
Let \mathcal{A} be a randomized external memory algorithm for the permutation problem on n integers of w bits each. Assume \mathcal{A} has error probability at most $1/3$ and let b denote the disk block size in number of bits. Let m denote the memory size measured in number of bits. Finally, let t denote the expected number of I/Os made by \mathcal{A} (on the worst input).

I/O-graphs. For an input array A representing a permutation π and bit strings d_1, \dots, d_n , and an output array C , define the (random) I/O-graph G of \mathcal{A} as follows: Initialize G to have one node per disk block in A and one node per disk block in C . Also, add one node to G representing the initial memory of \mathcal{A} . We think of the nodes representing the disk blocks of A and C as *block nodes* and the node representing the memory as a *memory node* (see Figure 1a). We will add more nodes and edges to G by observing the execution of \mathcal{A} on A . To simplify the description, we will call nodes of G either *dead* or *live*. We will always have at most one live memory node. Initially, all nodes are live. We use 0 to label the memory node. Moreover, we label the block nodes by consecutive integers starting at 1. Thus, the block nodes in the initial graph are labeled $1, 2, \dots, n(w + \lg n)/b + nw/b$.

Now, run algorithm \mathcal{A} on A . Whenever it makes an I/O, do as follows: If this is the first time, the block is being accessed and it is not part of the input or output (a write operation to an untouched block). Then, create a new live block node in G and add a directed edge from the current live memory node to the new block node (see Figure 1e). Label the new node by the next unused integer label. Otherwise, let v be the live node in G corresponding to the last time the disk block was accessed. We add a directed edge from v to the live memory node, mark v as dead, create a new live block node v' , and add a directed edge from the live memory node to v' . We give the new node the same label as v (Figure 1b and c). Finally, once for every m/b I/Os, we mark the memory node as dead, create a new live memory node, and add a directed edge from the old memory node to the new live memory node (Figure 1d).

To better understand the definition of G , observe that all the nodes with the same label represent the different versions of a disk block that existed throughout the execution of the algorithm. Moreover, there is always exactly one live node with any fixed label, representing the current version of

Figure 1. I/O-graph for an array A consisting of 3-bit strings d_1, \dots, d_b . In this example, each disk block contains two words of $w = 3$ bits, that is, $B = 2$ (and $b = Bw = 6$). Also, the main memory holds $M = 6$ words ($m = 18$). Figure (a) shows the initial I/O-graph. For each disk block, we have initially one block node that is illustrated underneath them. Black nodes are dead, and white nodes are live. Figure (b) shows the updated I/O-graph after making an I/O to access the first disk block. Figure (c) is the I/O-graph after accessing the block containing $C[1]$ and $C[2]$. Figure (d) shows the graph after making another I/O on the first disk block. Also, we create a new memory node after every $m/b = M/B = 3$ I/Os and mark the old memory node as dead. Figure (e) shows the updated graph after accessing some block other than the input or output.



the disk block. Also, observe that at the end of the execution, there must be a live disk block node in G representing each of the output blocks in C , and these have the same labels as the original nodes representing the empty disk blocks of C before the execution of \mathcal{A} .

Fixing the randomness of \mathcal{A} . Consider the execution of \mathcal{A} on an input A representing a uniform random permutation π as well as independent and uniform random bit strings $d_1, \dots, d_n \in \{0, 1\}^w$. Because \mathcal{A} makes an expected t I/Os, it follows by Markov's inequality that \mathcal{A} makes more than $6t$ I/Os with probability less than $1/6$. If we simply abort in such cases, we obtain an algorithm with worst case $O(t)$ I/Os and error probability at most $1/3 + 1/6 = 1/2$. Now fix the random choices of \mathcal{A} to obtain a deterministic algorithm \mathcal{A}^* with error probability $1/2$ over the random choice of π and d_1, \dots, d_n . \mathcal{A}^* makes t^* I/Os in the worst case. Observe that for \mathcal{A}^* , we get a fixed I/O graph $G(A)$ for every input array A because \mathcal{A}^* is deterministic.

Finding a popular I/O-graph. We now find an I/O-graph G that is the result of running \mathcal{A}^* on a large number of different inputs. For notational convenience, let t denote the worst case number of I/Os made by \mathcal{A}^* (instead of using t^* or $6t$). Observe that the total number of different I/O-graphs one can obtain as the result of running \mathcal{A}^* is small:

LEMMA 8. *There are no more than*

$$(t + n(w + \lg n)/b + nw/b + 1)^{t+1}$$

I/O-graphs that may result from the execution of \mathcal{A}^ .*

This means that we can find an I/O-graph, which corresponds to the execution of \mathcal{A}^* on many different inputs, and moreover, we can even assume that \mathcal{A}^* is correct on many such inputs:

LEMMA 9. *There exists a set Γ containing at least $(n!2^{nw})/(2(t + n(w + \lg n)/b + nw/b + 1)^{t+1})$ different input arrays A , such that \mathcal{A}^* is correct on all inputs $A \in \Gamma$ and the I/O-graph is the same for all $A \in \Gamma$.*

Data must travel far. The key idea in our lower bound proof is to argue that there is a permutation for which most data bit strings d_i are very far away from output entry $C[\pi(i)]$ in the corresponding I/O-graph. This would require the data to “travel” far. By Conjecture 1, this is impossible unless the I/O-graph is large. Thus, we start by arguing that there is a fixed permutation where data has to travel far on the average, and where it also holds that there are many different data values that can be sent using the same I/O-graph. To make this formal, let $\text{dist}(\pi, i, G)$ denote the distance between the block node in G representing the input block storing $A[i]$ (the initial node, before any I/Os were performed) and the node in G representing the output block storing $C[\pi(i)]$ in the *undirected* version of G (undirect all edges).

We prove the following:

LEMMA 10. *If $(t+n(w+\lg n)/b+nw/b+1)^{t+1} \leq (nw/b)^{n/30}$, then there exists a permutation π , a collection of values $\mathcal{F} \subseteq \{\{0, 1\}^w\}^n$ and an I/O-graph G such that the following holds:*

1. For all $(d_1, \dots, d_n) \in \mathcal{F}$, it holds that the algorithm \mathcal{A}^* executed on the input array A corresponding to inputs π and d_1, \dots, d_n results in the I/O-graph G and \mathcal{A}^* is correct on A .
2. $|\mathcal{F}| \geq \frac{2^{nw}}{4 + (t + n(w + \lg n) / b + nw / b + 1)^{t+1}}$.
3. There are at least $(4/5)n$ indices $i \in \{1, \dots, n\}$ for which $\text{dist}(\pi, i, G) \geq (1/2) \lg_{2m/b}(nw/b)$.

Reduction to network coding. We are now ready to make our reduction to network coding. The basic idea in our proof is to use Lemma 10 to obtain an I/O-graph G and permutation π with large distance between the node containing $A[i]$ and the node containing $C[\pi(i)]$ for many i . We will then create a source s_i at the node representing $A[i]$ and a corresponding sink t_i at the node corresponding to $C[\pi(i)]$. These nodes are far apart, but using the external memory permutation algorithm \mathcal{A}^* , there is an algorithm for transmitting d_i from s_i to t_i . Because the distance between s_i and t_i is at least $(1/2) \lg_{2m/b}(nw/b)$ for $(4/5)n$ of the pairs (s_i, t_i) , it follows from Conjecture 1 that the sum of capacities in the network must be at least $\Omega(nw \lg_{2m/b}(nw/b))$ (we can transmit w bits between each of the pairs). However, running the external memory algorithm results in a network/graph G with only $O(t)$ edges, each needing to transmit only b bits (corresponding to the contents of block on a read or write). Thus, each edge needs only have capacity b bits for the reduction to go through. Hence, the sum of capacities in the network is $O(tb)$. This means that $t = \Omega((nw/b) \lg_{2m/b}(nw/b))$ as desired.

However, the reduction is not as straightforward as that. The problem is that Lemma 10 leaves us only with a subset \mathcal{F} of all the possible values d_1, \dots, d_n that one wants to transmit. For other values of d_1, \dots, d_n , we cannot use the algorithm \mathcal{A}^* to transmit the data via the network/graph G . We could of course try to sample (d_1, \dots, d_n) uniformly from \mathcal{F} and then have a network coding solution only for such inputs. The problem is that for such a uniform $(d_1, \dots, d_n) \in \mathcal{F}$, it no longer holds that the inputs to the sources in the coding network are independent! Network coding rate only speaks of independent sources; hence, we need a way to break this dependency. We do this by adding an extra node u and some edges to the coding network. This extra node u serves as a coordinator that takes the independent sources X_1, \dots, X_n and replaces them with an input $(d_1, \dots, d_n) \in \mathcal{F}$ in such a way that running \mathcal{A}^* on (d_1, \dots, d_n) and using a little extra communication from u allow the sinks to recover $X_{\pi^{-1}(i)}$ from $d_{\pi^{-1}(i)}$. We proceed to give the formal construction. Let \tilde{G} be the I/O-graph, π the permutation, and $\mathcal{F} \subseteq \{\{0, 1\}^w\}^n$ the values promised by Lemma 10. From \tilde{G} , construct a coding network G^* as follows:

1. Add source and sink nodes s_1, \dots, s_n and t_1, \dots, t_n to G^* .
2. For each source s_i , add an additional node p_i .
3. Add all nodes of \tilde{G} to G^* .

4. Add all edges of \tilde{G} to G^* . Edges between a block node and a memory node have capacity b bits. Edges between two memory nodes have capacity m bits.
5. Remove all block nodes that have an incoming and outgoing edge to the same memory node (this makes the graph acyclic).
6. Add a directed edge with capacity w bits from each source s_i to p_i , and add a directed edge with capacity w bits from each p_i to the input block node containing $A[i]$.
7. Add an edge with capacity w bits from the output block node containing $C[\pi(i)]$ to the sink t_i .
8. Add a special node u to G^* . Add an edge of capacity w bits from each source s_i to u . Also, add a directed edge from u to each p_i having capacity ρ_i for parameters $\rho_i > 0$ to be fixed later. Also, add an edge from u to sink t_i with capacity ρ_i .

We argue that for sufficiently large choices of ρ_i , one can use \mathcal{A}^* to efficiently transmit w bits of information between every source-sink pair (s_i, t_i) . Our protocol for this problem uses Lemma 7 from Section 3 as a subroutine. We defer the proof of Lemma 7. It can be shown that there exists a transmitting protocol for transmitting X_1, \dots, X_n and it satisfies all capacity constraints of network G^* . The exact protocol can be found in the full version of the paper.

Deriving the lower bound. We observe that for all edges, except those with capacity ρ_i , our protocol sends a fixed number of bits. Thus, messages on such edges are prefix-free. For the edges with capacity ρ_i , the protocol sends a prefix-free message with expected length ρ_i . Because all messages on all edges are prefix-free, it follows from Shannon's Source Coding theorem that the expected length of each message is an upper bound on its entropy. Because the expected lengths are at most the capacity of the corresponding edges, we get by the definition of network coding rate from Section 2, that the above solution achieves a rate of w bits. Hence, from Conjecture 1, it follows that if we undirected G^* , then the multicommodity flow rate must be at least w bits. From the definition of multicommodity flow rate in Section 2, we see that this implies that there is a (possibly fractional) way of sending w units of flow between each source-sink pair.

We first examine the amount of flow that can be transported between pairs (s_i, t_i) along paths that visit u . We observe that any such flow must use at least two edges incident to u . But the sum of capacities of edges incident to u is $nw + 2 \sum_i \rho_i$. Hence, the amount of flow that can be transmitted along paths using u as an intermediate node is no more than $(nw + 2 \sum_i \rho_i) / 2 = nw / 2 + \sum_i \rho_i$. If $|\mathcal{F}| \geq 2^{nw - o(nw)}$, then this is no more than $nw / 2 + o(nw)$. From Lemma 10, we know that there are at least $(4/5)n$ indices i for which $\text{dist}(\pi, i, G) \geq (1/2) \lg_{2m/b}(nw/b)$, provided that $(t + n(w + \lg n) / b + nw / b + 1)^{t+1} \leq (nw/b)^{(1/30)n}$. The total flow that must be sent between such pairs is $(4/5)nw$. This means that there is at least $(4/5)nw - nw / 2 - o(nw) = \Omega(nw)$ flow that has to traverse $(1/2) \lg_{2m/b}(nw/b) = \Omega(\lg_{2m/b}(nw/b))$ edges of G^* (the flow must use a path in the undirected version of G as it cannot shortcut via u). Hence, the sum of capacities corresponding to edges in G must be $\Omega(nw \lg_{2m/b}(nw/b))$, assuming that

$|\mathcal{F}| \geq 2^{nw-o(nw)}$. Every I/O made by \mathcal{A}^* increases the capacity of the edges by $O(b)$ bits (two edges of b bit capacity when a new block node is added to G , and an amortized b bits capacity to pay for the m bit edge between memory nodes after every m/b I/Os). Thus, if \mathcal{A}^* makes at most t I/Os, it must be the case that $tb = \Omega(nw \lg_{2m/b}(nw/b))$ if $|\mathcal{F}| \geq 2^{nw-o(nw)}$. But $|\mathcal{F}| \geq 2^{nw/4(t+n(w+\lg n)/b+nw/b+1)^{t+1}}$. Therefore, we must have either $t = \Omega((nw/b) \lg_{2m/b}(nw/b))$ or $t \lg(tn(w+\lg n)/b) = \Omega(nw)$. Finally, Lemma 10 is also required $(t+n(w+\lg n)/b+nw/b+1)^{t+1} \leq (nw/b)^{(1/30)^n}$. Combining all of this means that for $t = \Omega((nw/b) \lg_{2m/b}(nw/b))$, or $t = \Omega(nw/\lg(nw))$ or $t = \Omega(n \lg(nw/b)/\lg(n \lg(nw/b))) = \Omega(n)$.

Thus, using the reduction to sorting we have proved:

$$w = \Omega(\lg n) \text{ either } t = \Omega((nw/b) \lg_{2m/b}(nw/b)), \text{ or}$$

$$t = \Omega(nw/\lg(nw)) = \Omega(n) \text{ or } t = \Omega(n \lg(nw/b)/\lg(n \lg(nw/b))) = \Omega(n).$$

For $w = \Omega(\lg n)$, we may use the reduction to sorting and we immediately obtain Theorem 2 as a corollary.

THEOREM 2. *Assuming Conjecture 1, any randomized algorithm for the external memory sorting problem with $w = \Omega(\lg n)$ bit integers, having error probability at most $1/3$, must make an expected*

$$\Omega\left(\min\left\{n, \frac{n}{B} \cdot \lg_{2M/B} \frac{n}{B}\right\}\right)$$

I/Os.

References

- Adler, M., Harvey, N.J.A., Jain, K., Kleinberg, R., Lehman, A.R. On the capacity of information networks. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, SODA '06* (2006), 241–250.
- Aggarwal, A., Vitter, J. The input/output complexity of sorting and related problems. *Commun. ACM* 9, 31 (1988), 1116–1127.
- Barak, B., Braverman, M., Chen, X., Rao, A. How to compress interactive communication. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing, STOC '10* (2010), 67–76.
- Braverman, M., Garg, S., Schwartzman, A. Coding in undirected graphs is either very helpful or not helpful at all. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9–11, 2017, Berkeley, CA, USA* (2017), 18:1–18:18.
- Han, Y. Deterministic sorting in $O(n \lg \lg n)$ time and linear space. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing* (2002), ACM, New York, 602–608.
- Han, Y., Thorup, M. Integer sorting in $O(n \sqrt{\lg \lg n})$ expected time and linear space. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science* (2002), IEEE, 135–144.
- Li, Z., Li, B. Network coding: the case of multiple unicast sessions. In *Proceedings of the 42nd Allerton Annual Conference on Communication, Control and Computing, Allerton '04* (2004).

Alireza Farhadi and Mohammad Taghi Hajiaghayi ([farhadi, hajiagha]@cs.umd.edu), University of Maryland, College Park, MD, USA.

Kasper Green Larsen (larsen@cs.au.dk), Aarhus University, Denmark.

Elaine Shi (runting@gmail.com), Cornell University, Ithaca, NY, USA.

© 2020 ACM 0001-0782/20/10 \$15.00

Digital Threats: Research and Practice (DTRAP)

Open for
Submissions

A peer-reviewed journal that targets the prevention, identification, mitigation, and elimination of digital threats



Digital Threats: Research and Practice (DTRAP) is a peer-reviewed journal that targets the prevention, identification, mitigation, and elimination of digital threats. DTRAP aims to bridge the gap between academic research and industry practice. Accordingly, the journal welcomes manuscripts that address extant digital threats, rather than laboratory models of potential threats, and presents reproducible results pertaining to real-world threats.

For further information and to submit your manuscript, visit dtrap.acm.org



CAREERS

Boston College

Non Tenure-Track Position in Computer Science

The Computer Science Department of Boston College seeks to fill one or possibly more non-tenure track teaching positions, as well as shorter-term visiting teaching positions. **One of these positions has a January, 2021 start date.** All applicants should be committed to excellence in undergraduate education and be able to teach a broad variety of undergraduate computer science courses. We are especially interested in candidates who are able to teach courses in systems and networks. Faculty in longer-term positions will also participate in the development of new courses that reflect the evolving landscape of the discipline.

Minimum requirements for the title of Assistant Professor of the Practice, and for the title of Visiting Assistant Professor, include a Ph.D. in Computer Science or closely related discipline.

Candidates without a Ph.D. would be eligible for the title of Lecturer or Visiting Lecturer.

We will begin reviewing applications as they are received and will continue considering

applications until the positions are filled. Applicants should submit a cover letter, CV, and a separate teaching statement and arrange for three confidential letters of recommendation that comment on their teaching performance to be uploaded directly to Interfolio. To apply go to: <http://apply.interfolio.com/78108>

Boston College conducts background checks as part of the hiring process. Information about the University and our department is available at bc.edu and cs.bc.edu.

Boston College is a Jesuit, Catholic university that strives to integrate research excellence with a foundational commitment to formative liberal arts education. We encourage applications from candidates who are committed to fostering a diverse and inclusive academic community. Boston College is an Affirmative Action/Equal Opportunity Employer and does not discriminate on the basis of any legally protected category including disability and protected veteran status. To learn more about how BC supports diversity and inclusion throughout the university, please visit the Office for Institutional Diversity at <http://www.bc.edu/offices/diversity>.

California Institute of Technology

Faculty Position in Computing and Mathematical Sciences

The Computing and Mathematical Sciences (CMS) Department at the California Institute of Technology (Caltech) invites applications for tenure-track faculty positions. The CMS Department is part of the Division of Engineering and Applied Science (EAS), comprising researchers working in and between the fields of aerospace, civil, electrical, environmental, mechanical, and medical engineering, as well as materials science and applied physics. The Institute as a whole represents the full range of research in biology, chemistry, engineering, geological and planetary sciences, physics, and the social sciences.

Fundamental research in computing and mathematical sciences, and applied research which links to activities in other parts of Caltech, are both welcomed. A commitment to world-class research, as well as high-quality teaching and mentoring, is expected, and appointment as an assistant professor is contingent upon the completion of a Ph.D. degree in applied mathematics, computer science or related areas. The initial appointment at the assistant professor level is four years. Reappointment beyond the initial term is contingent upon successful review conducted prior to the commencement of the fourth year.

► Interviews will take place in January and February 2021.

► Applications will be reviewed beginning 22 October 2020 and all applications received before 1 December 2020 will receive full consideration.

► Applications received before 8 November will be considered for interviews in January.

► Applications received after 8 November will be considered for interviews in February.

To fulfill Caltech's commitment to promoting diversity, inclusiveness, and excellence in research on our campus, we actively seek candidates who can work with, teach, and mentor students from under-represented communities. Along with other standard application materials, applicants should submit a diversity and inclusion statement that discusses past and/or anticipated contributions to improving diversity, equity, and inclusion in the areas of research, teaching, and/or outreach.

For a list of all documents required, and full instructions on how to apply online, please visit <https://applications.caltech.edu/jobs/cms>. Questions about the application process may be directed to search@cms.caltech.edu.

Caltech is an equal opportunity employer and all qualified applicants will receive consideration for employment without regard to age, race, color, religion, sex, sexual orientation, gender identity, national origin, disability status, protected veteran status, or any other characteristic protected by law.



TENURE-TRACK AND TENURED POSITIONS School of Information Science and Technology (SIST)

ShanghaiTech University invites highly qualified candidates to fill multiple tenure-track/tenured faculty positions as its core founding team in the School of Information Science and Technology (SIST). We seek candidates with exceptional academic records or demonstrated strong potentials in all cutting-edge research areas of information science and technology. They must be fluent in English. English-based overseas academic training or background is highly desired.

ShanghaiTech is founded as a world-class research university for training future generations of scientists, entrepreneurs, and technical leaders. Boasting a new modern campus in Zhangjiang Hightech Park of cosmopolitan Shanghai, ShanghaiTech shall trail-blaze a new education system in China. Besides establishing and maintaining a world-class research profile, faculty candidates are also expected to contribute substantially to both graduate and undergraduate educations.

Academic Disciplines: Candidates in all areas of information science and technology shall be considered. Our recruitment focus includes, but is not limited to: computer science and technology, electronic science and technology, information and communication engineering, applied mathematics and statistics, data science, robotics, bioinformatics, biomedical engineering, internet of things, smart energy, computer systems and security, operation research, mathematical optimization and other interdisciplinary fields involving information science and technology, especially areas related to AI.

Compensation and Benefits: Salary and startup funds are highly competitive, commensurate with experience and academic accomplishment. We also offer a comprehensive benefit package to employees and eligible dependents, including on-campus housing. All regular ShanghaiTech faculty members will join its new tenure-track system in accordance with international practice for progress evaluation and promotion.

Qualifications:

- Strong research productivity and demonstrated potentials;
- Ph.D. (Electrical Engineering, Computer Engineering, Computer Science, Statistics, Applied Math, or related field);
- A minimum relevant (including PhD) research experience of 4 years.

Applications: Submit (in English, PDF version) a cover letter, a 2-page research plan, a CV plus copies of 3 most significant publications, and names of three referees to: sist@shanghaitech.edu.cn

For more information, please visit: <http://sist.shanghaitech.edu.cn/>

Deadline: December 31, 2020



Association for
Computing Machinery

Digital Government: Research and Practice (DGOV)

*An Open Access research journal on the
potential and impact of technology on
governance innovations and public institutions*

Digital Government: Research and Practice (DGOV) is an interdisciplinary journal on the potential and impact of technology on governance innovations and its transformation of public institutions. It promotes applied and empirical research from academics, practitioners, designers, and technologists, using political, policy, social, computer, and data sciences methodologies.

DGOV aims to appeal to a wider audience of research and practice communities with novel insights, disruptive design ideas, technical solutions, scientific and empirical knowledge, and a deep understanding of digital impact in the public sector. The major areas include the new forms of governance and citizen roles in the inter-connected digital environment, as well as the governance of new technologies, including governance of automation, sensor devices, robot behavior, artificial intelligence, and big data. Whether it is governing technology or technology for governing, the goal is to offer cutting-edge research and concepts designed to navigate and balance the competing demands of transparency and cybersecurity, innovation and accountability, and collaboration and privacy.



For further information and to submit
your manuscript, visit dgov.acm.org

[CONTINUED FROM P. 108] approach is that it requires doubling the number of people polled to get the same effective sample size. In terms of this example, 200 people must be sampled to get an effective sample size of 100.

Question: Suppose T has the support of approximately 60% of the people and B has the support of 40%. Suppose the stigma is against only B supporters. Assuming the people who are polled are given a standard deck of 52 playing cards, can you make it so that if a person responds B, then that person has approximately a 50% chance of actually supporting B and achieve an effective sample size of 100 by polling only 140 people?

Solution: Suppose that with probability $2/7$ a person will say B regardless of his or her view. Then if 60% want T and 40% want B, B will receive $(2/7) \cdot 140 = 40$ votes because of this $2/7$ probability and another 40 from the committed B supporters. Such a scheme would have the goal of having anyone who responds with B to the pollster to have a 50% chance of supporting B. To achieve this, the pollster says to the pollee: "Please take seven cards including an Ace, 2, 3, 4, 5, 6, and 7, regardless of suit. Shuffle the seven cards. Now turn over one card. If it is an Ace or a 7, say B. Otherwise, please tell me what you really think." In this way, only 40 extra people must be interviewed to get an effective sample size of 100 or in general $2/7$ extra people.

Upstart: Generalize the above solution to k candidates all with approximately equal support. Each person should have a probability of no more than p of actually supporting the candidate he or she mentions. You can assume for this purpose the pollee has access to a trusted random number generator that will give a number between 0 and 1 with uniform probability. It should be enough to use this random number generator just once per pollee.

Dennis Shasha (dennisshasha@yahoo.com) is a professor of computer science in the Computer Science Department of the Courant Institute at New York University, New York, USA, as well as the chronicler of his good friend the omniheurist Dr. Ecco.

All are invited to submit their solutions to upstartpuzzles@cacm.acm.org; solutions to upstarts and discussion will be posted at <http://cs.nyu.edu/cs/faculty/shasha/papers/cacmpuzzles.html>

Copyright held by author.



DOI:10.1145/3416266

Dennis Shasha

Upstart Puzzles

Privacy-Preserving Polling

Can you answer a poll without revealing your true preferences and have the results of the poll still be accurate?

WHEN PEOPLE ARE asked whom they will vote for, they might not want to say. After all, other people might judge them, ask for contributions, or publish the answer. Suppose there are two candidates, randomly called B and T. Suppose, again for the sake of this hypothetical, that there is a slight stigma against people who support T.

The pollster says to them: “Please flip a coin. If the coin comes up tails, please tell us whom you like best. If it comes up heads, then always say T.” That way, even if a person states an intention to vote for T, nobody knows for sure.

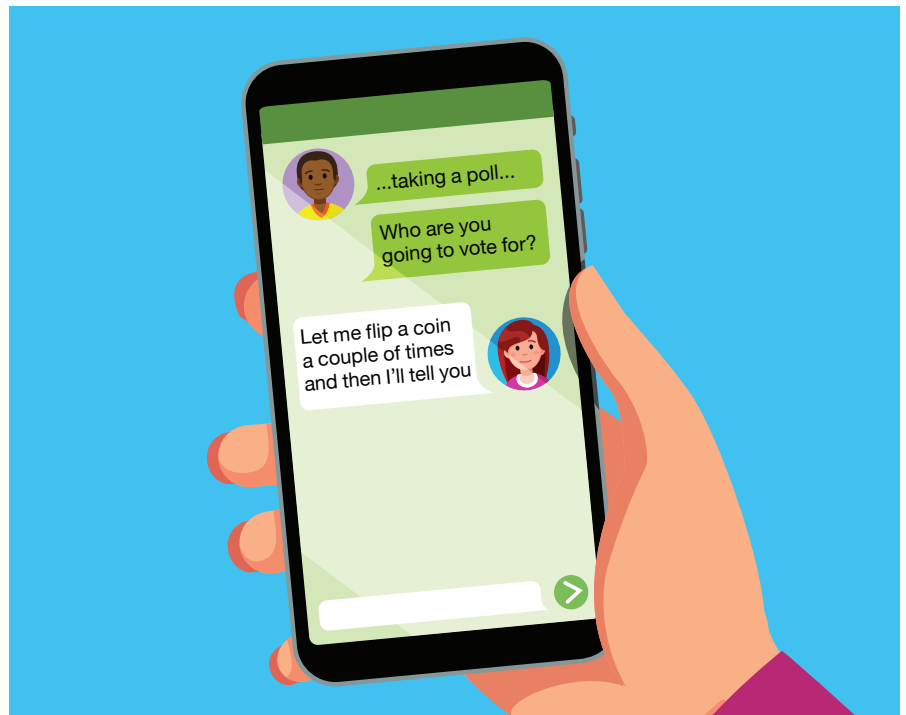
Warm-Up: Suppose the true probabilities are 60% for B and 40% for T. 200 people are again polled. How many will say T in response to the poll with the coin-flip rule and how many will say B?

Solution to Warm-Up: Approximately half the people—100—will flip heads and will say T, regardless of their preferences. Of the other half—60—will say B and 40 will say T. So 140 will say T and 60 will say B.

Warm-Up 2: Suppose 70% want T and 30% want B. 200 people are again polled. How many will say T in response to the poll with the coin-flip rule and how many will say B?

Solution to Warm-Up 2: 170 for T and 30 for B. So to find the true support for T and B, simply subtract from the T score half of the total number of people polled. Leave the B score alone.

But now suppose a country is so divided that, depending on whom you



Political pollster: “We understand your choice of candidate may be something you want to keep private. At the end of this process, only you will know for sure whether the choice you mention is your real choice or not.”

talk to, there might be a stigma to vote for either candidate. Can the pollsters still do their job?

Question: Can you think of a protocol that will protect privacy for supporters both of B and T?

Solution: Here is one possibility. Tell the pollees (the people asked): “Please flip a coin twice. If it comes up heads both times, then please say T. If it comes up tails both times, please say B. With any other combination, please

tell us the truth.” Suppose again for the purposes of example 60% want B and 40% want T. If 200 people are polled, B will get 60 true answers and 50 because of double tails. The remaining 90 will go to T. Thus, we subtract a quarter of the total number of people polled (50 in this example) from B (yielding $110 - 50 = 60$), a quarter from T (yielding $90 - 50 = 40$) and we get the correct answer.

The only trouble with this privacy-preserving [CONTINUED ON P. 107]

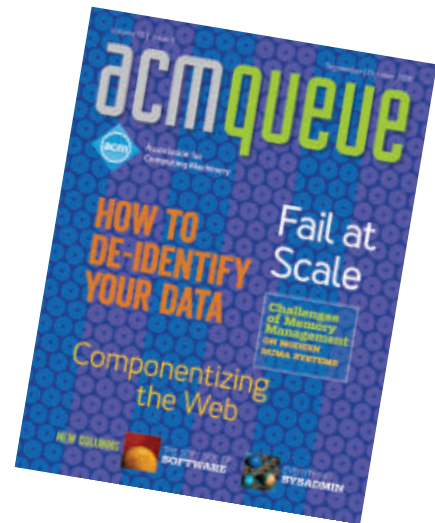
acmqueue

Check out the acmqueue app

FREE TO ACM MEMBERS

acmqueue is ACM's magazine by and for practitioners, bridging the gap between academics and practitioners of the art of computer science. For more than a decade *acmqueue* has provided unique perspectives on how current and emerging technologies are being applied in the field, and has evolved into an interactive, socially networked, electronic magazine.

Broaden your knowledge with technical articles focusing on today's problems affecting CS in practice, video interviews, roundtables, case studies, and lively columns.

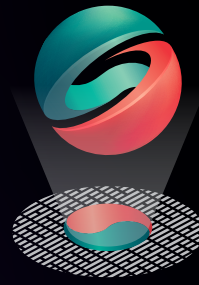


Keep up with this fast-paced world on the go. Download the mobile app.



Association for
Computing Machinery

Desktop digital edition also available at queue.acm.org.
Bimonthly issues free to ACM Professional Members.
Annual subscription \$19.99 for nonmembers.



SIGGRAPH ASIA 2020 DAEGU

The 13th ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia

Conference 17 – 20 November 2020

Exhibition 18 – 20 November 2020

EXCO, Daegu, South Korea

Driving Diversity

SA2020.SIGGRAPH.ORG

[#SIGGRAPHAsia](https://twitter.com/SIGGRAPHAsia) | [#SIGGRAPHAsia2020](https://twitter.com/SIGGRAPHAsia2020)



Sponsored by



Organized by

