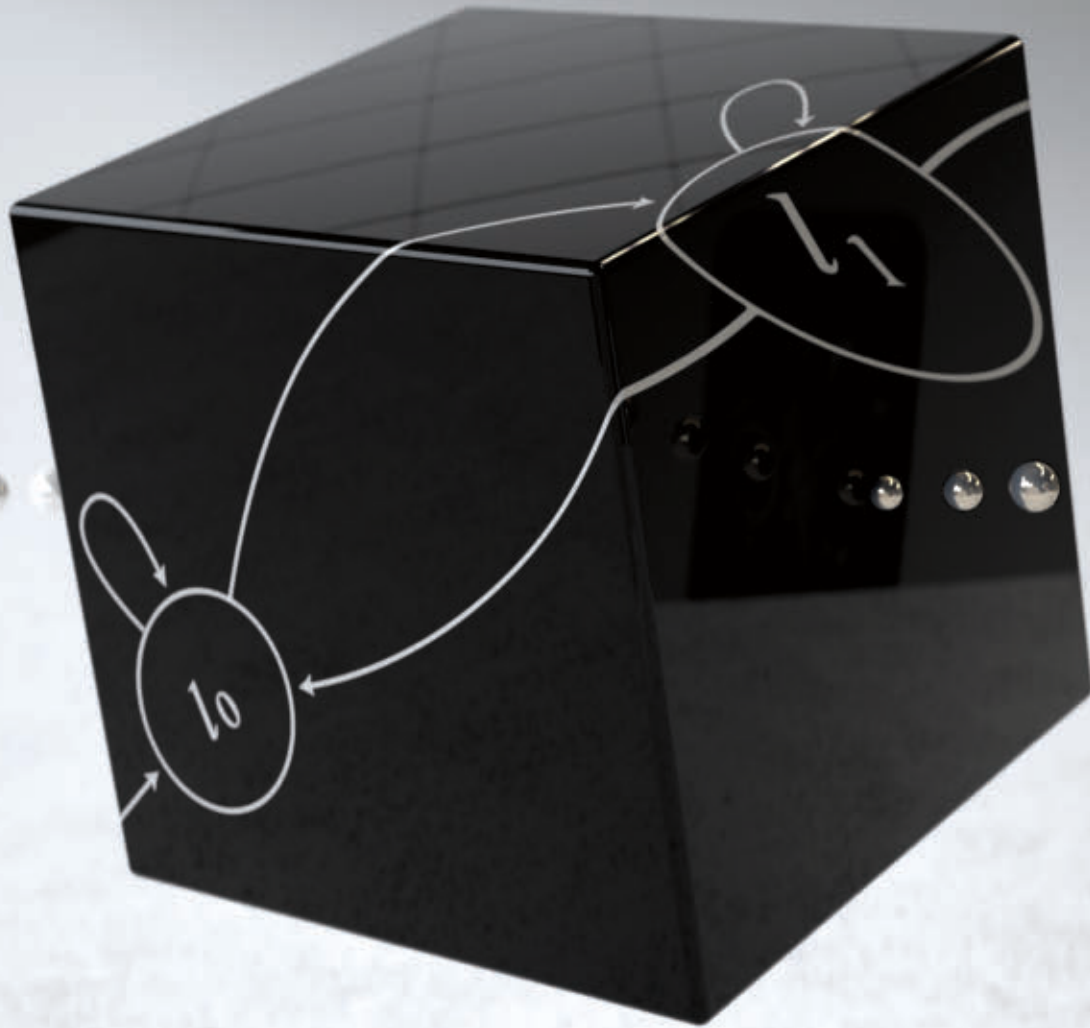


COMMUNICATIONS

CACM.ACM.ORG OF THE ACM 02/2017 VOL.60 NO.02



Model Learning

*The power—and potential—
of black box techniques*

Copyright Enforcement in the Digital Age

Life Beyond Distributed Transactions

Are Computer Chips the New Security Threat?

Social and Ethical Behavior in the Internet of Things

CELEBRATING 50 YEARS OF COMPUTING'S GREATEST ACHIEVEMENTS

Since its inauguration in 1966, the ACM A. M. Turing Award has recognized major contributions of lasting importance in computing. Through the years, it has become the most prestigious technical award in the field, often referred to as the "Nobel Prize of computing."

ACM will celebrate 50 years of the Turing Award and the visionaries who have received it with a conference on June 23 - 24, 2017 at the Westin St. Francis in San Francisco. ACM Turing laureates will join other ACM award recipients and experts in moderated panel discussions exploring how computing has evolved and where the field is headed. Topics include:

- **Advances in Deep Neural Networks**
- **Restoring Personal Privacy without Compromising National Security**
- **Moore's Law Is Really Dead: What's Next?**
- **Quantum Computing: Far Away? Around the Corner? Or Maybe Both at the Same Time?**
- **Challenges in Ethics and Computing**
- **Preserving Our Past for the Future**
- **Augmented Reality: From Gaming to Cognitive Aids and Beyond**

We hope you can join us in San Francisco, or via our live web stream, to look ahead to the future of technology and innovation, and to help inspire the next generation of computer scientists to invent and dream.

For more information and to reserve your spot, visit www.acm.org/turing-award-50

Program Committee

Craig Partridge
Program Chair

Fahad Dogar
Deputy Program Chair

Karen Breitman

Vint Cerf

Jeff Dean

Joan Feigenbaum

Wendy Hall

Joseph Konstan

David Patterson



CELEBRATING 50 YEARS
OF COMPUTING'S GREATEST ACHIEVEMENTS

Systor2017

May 22 – 24 Haifa, Israel

10 The 10th ACM International Systems and Storage Conference

Paper submission deadline: February 22, 2017

Paper acceptance notification: March 29, 2017

Poster submission deadline: March 22, 2017

We invite you to submit original and innovative papers, covering all aspects of computer systems technology, such as file and storage technology; operating systems; distributed, parallel, and cloud systems; security; virtualization; and fault tolerance, reliability, and availability. SYSTOR 2017 accepts full papers, short papers, and posters.

Program chairs

Peter Desnoyers, Northeastern University

Eyal de Lara, University of Toronto

General chair

Doron Chen, IBM Research

Posters chair

Adam Morrison, Tel Aviv University

Steering committee head

Michael Factor, IBM Research

Steering committee

Ethan Miller, University of California Santa Cruz

Liuba Shrira, Brandeis University

Dan Tsafir, Technion

Dalit Naor, IBM Research

Erez Zadok, Stony Brook University

www.systor.org/2017/

Sponsored by



In cooperation with



Platinum sponsor



Gold sponsors



NOKIA Bell Labs

Sponsors



ORACLE



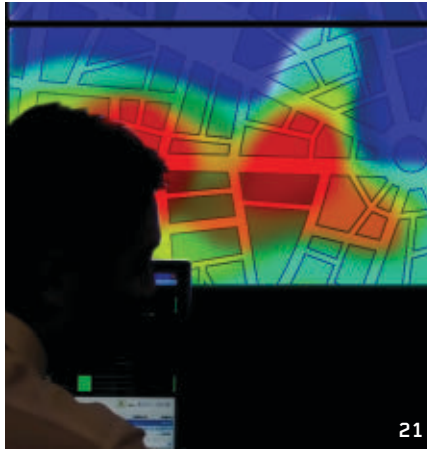
Departments

- 5 **From the President**
Celebrating 50 Years of the Turing Award
By Vicki L. Hanson
-
- 6 **Cerf's Up**
Social and Ethical Behavior in the Internet of Things
By Francine Berman and Vinton G. Cerf
-
- 8 **Letters to the Editor**
Use the Scientific Method in Computer Science
-
- 10 **Panels in Print**
Artificial Intelligence
-
- 12 **BLOG@CACM**
Liberal Arts Academia Wants YOU!
Janet Davis makes a plea to CS practitioners to consider even a short teaching stint.
-
- 41 **Calendar**
-
- 116 **Careers**

Last Byte

- 120 **Future Tense**
Fatal Guidance
In a series of interactive murder mysteries, I might not have done it, but, then again, maybe I did.
By William Sims Bainbridge

News



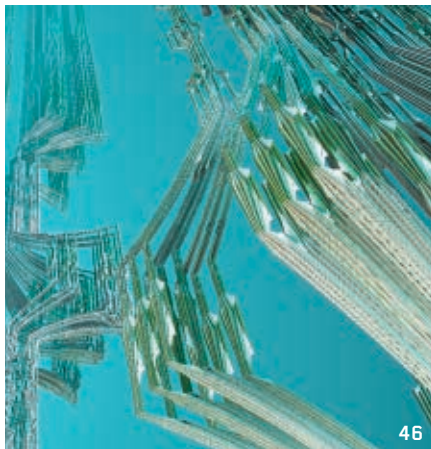
- 15 **Secure Quantum Communications**
Data locking experiments provide stepping stones to a possible future in quantum cryptography.
By Chris Edwards
-
- 18 **Are Computer Chips the New Security Threat?**
Security researchers have identified a technique for installing a backdoor on computer chips, a security flaw that could profoundly change the computing industry.
By Samuel Greengard
-
- 21 **It's Not the Algorithm, It's the Data**
In risk assessment and predictive policing, biased data can yield biased results.
By Keith Kirkpatrick

Viewpoints

- 26 **Inside Risks**
The Future of the Internet of Things
The IoT can become ubiquitous worldwide—if the pursuit of systemic trustworthiness can overcome the potential risks.
By Ulf Lindqvist and Peter G. Neumann
-
- 31 **Education**
Fostering Creativity through Computing
How creative thinking tools and computing can be used to support creative human endeavors.
By Aman Yadav and Steve Cooper
-
- 34 **Kode Vicious**
The Unholy Trinity of Software Development
Tests, documentation, and code.
By George V. Neville-Neil
-
- 37 **Privacy and Security**
User-Centric Distributed Solutions for Privacy-Preserving Analytics
How can cryptography empower users with sensitive data to access large-scale computing platforms in a privacy-preserving manner?
By Azer Bestavros, Andrei Lapets, and Mayank Varia
-
- 40 **Viewpoint**
Smart Machines Are Not a Threat to Humanity
Worrying about machines that are too smart distracts us from the real and present threat from machines that are too dumb.
By Alan Bundy
-
- 43 **Viewpoint**
AI Dangers: Imagined and Real
Arguing against the arguments for the concept of the singularity.
By Devdatt Dubhashi and Shalom Lappin



Practice



46

46 **Life Beyond Distributed Transactions**
An apostate's opinion.
By Pat Helland

55 **Are You Load Balancing Wrong?**
Anyone can use a load balancer.
Using it properly is much more difficult.
By Thomas A. Limoncelli

58 **BBR: Congestion-Based Congestion Control**
Measuring bottleneck bandwidth and round-trip propagation time.
By Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson

Q Articles' development led by **acmqueue**
queue.acm.org



About the Cover:
Model learning is emerging as an effective method for achieving black box state machine models of software and hardware components. Frits Vaandrager explores the latest progress and applications in the field on p. 86. Cover illustration by Marie Dommenget.

Contributed Articles



76

68 **Copyright Enforcement in the Digital Age: Empirical Evidence and Policy Implications**
Government-sanctioned and market-based anti-piracy measures can both mitigate economic harm from piracy.
By Brett Danaher, Michael D. Smith, and Rahul Telang

76 **Computing History Beyond the U.K. and U.S.: Selected Landmarks from Continental Europe**
It is past time to acknowledge 400 years of European computational innovation from non-English-speaking scientists and engineers.
By Herbert Bruderer



Watch the author discuss his work in this exclusive *Communications* video.
<http://cacm.acm.org/videos/computing-history-beyond-the-uk-and-us>

Review Articles



86

86 **Model Learning**
Model learning emerges as an effective method for black-box state machine models of hardware and software components.
By Frits Vaandrager



Watch the author discuss his work in this exclusive *Communications* video.
<http://cacm.acm.org/videos/model-learning>

Research Highlights

98 **Technical Perspective**
Cleaning Up Flaws in TLS Implementations
By Eric Rescorla

99 **A Messy State of the Union: Taming the Composite State Machines of TLS**
By Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue

108 **Authentication Using Pulse-Response Biometrics**
By Ivan Martinovic, Kasper Rasmussen, Marc Roeschlin, and Gene Tsudik



ACM, the world's largest educational and scientific computing society, delivers resources that advance computing as a science and profession. ACM provides the computing field's premier Digital Library and serves its members and the computing profession with leading-edge publications, conferences, and career resources.

Executive Director and CEO
Bobby Schnabel
Deputy Executive Director and COO
Patricia Ryan
Director, Office of Information Systems
Wayne Graves
Director, Office of Financial Services
Darren Ramdin
Director, Office of SIG Services
Donna Cappo
Director, Office of Publications
Scott E. Delman

ACM COUNCIL

President
Vicki L. Hanson
Vice-President
Cherri M. Pancake
Secretary/Treasurer
Elizabeth Churchill
Past President
Alexander L. Wolf
Chair, SGB Board
Jeanna Matthews
Co-Chairs, Publications Board
Jack Davidson and Joseph Konstan
Members-at-Large
Gabrielle Anderst-Kotis; Susan Dumais; Elizabeth D. Mynatt; Pamela Samuelson; Eugene H. Spafford
SGB Council Representatives
Paul Beame; Jenna Neefe Matthews; Barbara Boucher Owens

BOARD CHAIRS

Education Board
Mehran Sahami and Jane Chu Prey
Practitioners Board
Terry Coatta and Stephen Ibaraki

REGIONAL COUNCIL CHAIRS

ACM Europe Council
Dame Professor Wendy Hall
ACM India Council
Srinivas Padmanabhuni
ACM China Council
Jianguang Sun

PUBLICATIONS BOARD

Co-Chairs
Jack Davidson; Joseph Konstan
Board Members
Ronald F. Boisvert; Karin K. Breitman; Terry J. Coatta; Anne Condon; Nikil Dutt; Roch Guerin; Carol Hutchins; Yannis Ioannidis; Catherine McGeoch; M. Tamer Ozsu; Mary Lou Soffa; Alex Wade; Keith Webster

ACM U.S. Public Policy Office
Renee Dopplick, Director
1828 L Street, N.W., Suite 800
Washington, DC 20036 USA
T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association
Mark R. Nelson, Executive Director

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

Communications of the ACM is the leading monthly print and online magazine for the computing and information technology fields. *Communications* is recognized as the most trusted and knowledgeable source of industry information for today's computing professional. *Communications* brings its readership in-depth coverage of emerging areas of computer science, new trends in information technology, and practical applications. Industry leaders use *Communications* as a platform to present and debate various technology implications, public policies, engineering challenges, and market trends. The prestige and unmatched reputation that *Communications of the ACM* enjoys today is built upon a 50-year commitment to high-quality editorial content and a steadfast dedication to advancing the arts, sciences, and applications of information technology.

STAFF

DIRECTOR OF PUBLICATIONS
Scott E. Delman
cacm-publisher@cacm.acm.org

Executive Editor
Diane Crawford
Managing Editor
Thomas E. Lambert
Senior Editor
Andrew Rosenbloom
Senior Editor/News
Larry Fisher
Web Editor
David Roman
Rights and Permissions
Deborah Cotton

Art Director
Andrij Borys
Associate Art Director
Margaret Gray
Assistant Art Director
Mia Angelica Balaquiot
Designer
Iwona Usakiewicz
Production Manager
Lynn D'Addesio
Advertising Sales Account Manager
Ilia Rodriguez

Columnists
David Anderson; Phillip G. Armour;
Michael Cusumano; Peter J. Denning;
Mark Guzdial; Thomas Haigh;
Leah Hoffmann; Mari Sako;
Pamela Samuelson; Marshall Van Alstyne

CONTACT POINTS

Copyright permission
permissions@hq.acm.org
Calendar items
calendar@cacm.acm.org
Change of address
acmhlp@acm.org
Letters to the Editor
letters@cacm.acm.org

WEBSITE
<http://cacm.acm.org>

AUTHOR GUIDELINES
<http://cacm.acm.org/>

ACM ADVERTISING DEPARTMENT

2 Penn Plaza, Suite 701, New York, NY
10121-0701
T (212) 626-0686
F (212) 869-0481

Advertising Sales Account Manager
Ilia Rodriguez
ilia.rodriguez@hq.acm.org

For display, corporate/brand advertising:
Craig Pitcher
pitcherc@acm.org T (408) 778-0300

Media Kit acmm mediasales@acm.org

Association for Computing Machinery (ACM)
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA
T (212) 869-7440; F (212) 869-0481

EDITORIAL BOARD

EDITOR-IN-CHIEF
Moshe Y. Vardi
eic@cacm.acm.org

NEWS

Co-Chairs
William Pulleyblank and Marc Snir
Board Members
Mei Kobayashi; Michael Mitzenmacher;
Rajeev Rastogi; François Sillion

VIEWPOINTS

Co-Chairs
Tim Finin; Susanne E. Hambrusch;
John Leslie King
Board Members
William Aspray; Stefan Bechtold;
Michael L. Best; Judith Bishop;
Stuart I. Feldman; Peter Freeman;
Mark Guzdial; Rachelle Hollander;
Richard Ladner; Carl Landwehr;
Carlos Jose Pereira de Lucena;
Beng Chin Ooi; Loren Terveen;
Marshall Van Alstyne; Jeannette Wing

Q PRACTICE

Co-Chair
Stephen Bourne
Board Members
Eric Allman; Peter Bailis; Terry Coatta;
Stuart Feldman; Benjamin Fried;
Pat Hanrahan; Tom Killalea; Tom Limoncelli;
Kate Matsudaira; Marshall Kirk McKusick;
George Neville-Neil; Theo Schlossnagle;
Jim Waldo

The Practice section of the CACM Editorial Board also serves as the Editorial Board of [queue](http://www.acm.org/queue).

CONTRIBUTED ARTICLES

Co-Chairs
Andrew Chien and James Larus
Board Members
William Aiello; Robert Austin; Elisa Bertino;
Gilles Brassard; Kim Bruce; Alan Bundy;
Peter Buneman; Peter Druschel; Carlo Ghezzi;
Carl Gutwin; Yannis Ioannidis;
Gal A. Kaminka; James Larus; Igor Markov;
Gail C. Murphy; Bernhard Nebel;
Lionel M. Ni; Kenton O'Hara; Sriram Rajamani;
Marie-Christine Rousset; Avi Rubin;
Krishan Sabnani; Ron Shamir; Yoav Shoham; Larry Snyder; Michael Vitale;
Wolfgang Wahlster; Hannes Werthner;
Reinhard Wilhelm

RESEARCH HIGHLIGHTS

Co-Chairs
Azer Bestavros and Gregory Morrisett
Board Members
Martin Abadi; Amr El Abbadi; Sanjeev Arora;
Michael Backes; Nina Balcan; Dan Boneh;
Andrei Broder; Doug Burger; Stuart K. Card;
Jeff Chase; Jon Crowcroft; Alexei Efros;
Alon Halevy; Norm Jouppi; Andrew B. Kahng;
Sven Koenig; Xavier Leroy; Steve Marschner;
Kobbi Nissim; Guy Steele, Jr.; David Wagner;
Margaret H. Wright; Nikolai Zeldovich;
Andreas Zeller

WEB Chair

James Landay
Board Members
Marti Hearst; Jason I. Hong;
Jeff Johnson; Wendy E. MacKay

ACM Copyright Notice

Copyright © 2017 by Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to publish from permissions@hq.acm.org or fax (212) 869-0481.

For other copying of articles that carry a code at the bottom of the first or last page or screen display, copying is permitted provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center; www.copyright.com.

Subscriptions

An annual subscription cost is included in ACM member dues of \$99 (\$40 of which is allocated to a subscription to *Communications*); for students, cost is included in \$42 dues (\$20 of which is allocated to a *Communications* subscription). A nonmember annual subscription is \$269.

ACM Media Advertising Policy

Communications of the ACM and other ACM Media publications accept advertising in both print and electronic formats. All advertising in ACM Media publications is at the discretion of ACM and is intended to provide financial support for the various activities and services for ACM members. Current advertising rates can be found by visiting <http://www.acm-media.org> or by contacting ACM Media Sales at (212) 626-0686.

Single Copies

Single copies of *Communications of the ACM* are available for purchase. Please contact acmhlp@acm.org.

COMMUNICATIONS OF THE ACM

(ISSN 0001-0782) is published monthly by ACM Media, 2 Penn Plaza, Suite 701, New York, NY 10121-0701. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER

Please send address changes to *Communications of the ACM*
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA

Printed in the U.S.A.



Association for
Computing Machinery



Celebrating 50 Years of the Turing Award

FIFTY YEARS AGO, ACM awarded the first A.M. Turing Award to Alan Perlis for his work on advanced programming techniques and compiler construction. Since then, the award has been given annually, with the 50th Turing Award presented last June to Whitfield Diffie and Martin Hellman for their work on public key encryption. In total, 64 men and women from around the world have received the Turing Award, recognizing work laying down the foundations of modern computing.

The prominence of the ACM Turing Award matches the impact of the contributions it honors. In the 50 years since its inception, the Award has become known as the “Nobel Prize of Computing.” Thanks to the generous support of Google, the award currently carries a \$1 million prize.

To celebrate the first 50 years of the Turing Award, ACM is sponsoring a year-long series of programs, as highlighted on the Turing 50th website <http://www.acm.org/turing-award-50>. This site consolidates information about the Turing Laureates and Alan Turing himself. It also presents the *Panels in Print* and provides information about the upcoming Turing 50th conference.

Panels in Print is a series of writings on key computing topics of the day. The first of these panels features Raj Reddy, Jeff Dean, David Blei, and Pedro Felzenszwalb discussing the state of artificial intelligence and can be found on p. 10 of this issue.

The culminating event of this anniversary year will be ACM’s Celebration of 50 Years of the Turing Award conference that will take place June 23–24, 2017, in San Francisco. This event will recognize achievements in computing and will hon-

or the Turing Laureates, providing opportunities to hear from many of them. For students and early career members of the computing community there also will be opportunities to meet and converse with the Laureates.

The conference program has been organized around seven moderated panel discussions designed to span a range of computing areas. The goal is to review topics of current interest to both those in the profession as well as to society at large. Those participating in these discussions will include Turing Laureates, ACM award winners, and others involved in shaping the direction of computing. We thank our distinguished Program Committee (Craig Partridge, Fahad Dogar, Karen Breitman, Vint Cerf, Jeff Dean, Joan Feigenbaum, Wendy Hall, Joseph Konstan, and David Patterson) who guided the choice of topics, moderators, and panelists. Panel topics will include:

► *Advances in Deep Neural Networks*: How are deep neural networks changing our world and our jobs and what breakthroughs may we imagine going forward?

► *Restoring Personal Privacy without Compromising National Security*: Can computing technology promote both personal privacy and national security?

► *Moore’s Law Is Really Dead: What’s Next?* What old doors will this seismic change close and what new doors will it open?

► *Quantum Computing: Far Away? Around the Corner? Or Maybe Both at the Same Time?* For both theory and practice, where we are headed, and what quantum skills might be needed by future computing professionals?

► *Challenges in Ethics and Computing*: How do we recognize and address ethical issues that arise with advances in technology?

► *Preserving our Past for the Future*: How do we archive our electronic artifacts to ensure we can read data and documents in both the near and distant future?

► *Augmented Reality—From Gaming to Cognitive Aids and Beyond*: How can the sensing and sensory display technologies of augmented reality empower individuals and communities?

There will be multiple opportunities to experience this Turing 50th Celebration event. If you can attend in person you may register at <http://www.acm.org/awards/turing-award-50-conference>. There is no registration fee for the meeting, but space will be limited so early registration is essential. Also, I would like to extend a special thank you to several of the SIGs who have sponsored (SIGARCH, SIGCHI, SIGCOMM, SIGGRAPH, SIGHPC, SIGIR, SIGKDD, SIGMM, SIGMOD, SIGPLAN, and SIGSOFT) and supported (SIGACCESS, SIGAI, SIGITE) the Turing event, including funding for students from their SIGs to attend.

If you are not able to attend the event in person, please note that the panel discussions will be streamed live. They also will be video recorded and made available (with subtitles/closed captioning) through the ACM website.

We hope you are able to experience this special ACM activity highlighting the range and impact of Turing Award-winning work. Whether it be reading the *Panels in Print*, watching the conference on video, or participating in person in San Francisco, ACM is working to make this a valuable experience. ■

Vicki L. Hanson (vlh@acm.org) is ACM President, Distinguished Professor at Rochester Institute of Technology, and a professor at the University of Dundee. Twitter: @ACM_President

Copyright held by author.

Social and Ethical Behavior in the Internet of Things

LAST OCTOBER, MILLIONS of interconnected devices infected with malware mounted a “denial-of-service” cyberattack on Dyn, a company that operates part of the Internet’s directory service. Such attacks require us to up our technical game in Internet security and safety. They also expose the need to frame and enforce social and ethical behavior, privacy, and appropriate use in Internet environments.

Social behavior and appropriate use become even more crucial as we build out the “Internet of Things” (IoT)—an increasingly interconnected cyber-physical-biological environment that links devices, systems, data, and people. At its best, the IoT has the potential to create an integrated ecosystem that can respond to a spectrum of needs, increasing efficiency and opportunity, and empowering people through technology, and technology through intelligence. At its worst, the IoT can open a Pandora’s Box of inappropriate and unsafe behavior, unintended consequences, and intrusiveness.

The difference between an IoT that enhances society and one that diminishes it will be determined by our ability to create an effective model for IoT governance. This model must guide social behavior and ethical use of IoT technologies while promoting effective security and safety. While we should not limit technology innovation too early with overly restrictive policy, neither should we leave the policy and governance discussion until the IoT is so mature that it cannot easily incorporate protections.

What Policy Will Be Needed for the IoT?

Although much of the policy needed for the IoT may evolve from Internet governance, the scale, heterogeneity, complexity, and degree of technological autonomy within the IoT will require new thinking about regulation and policy and force new interpretations of current law. As an example of the complexity of the governance challenge, consider three key areas critical to ensure the positive potential of the IoT:

1. What are your rights to privacy in the IoT? The IoT will sharpen the tension between individual privacy and the use of personal information to promote effectiveness, safety, and security. Who should control information about you? Who should access it? Who can use it? The answer is not always clear-cut. Consider medical monitoring devices and the information they accumulate. Should your personal health information be shared when

We need to be thinking deeply now about broad IoT use and deployment, and how it can help create a more enlightened and civilized society.

the Centers for Disease Control want to track a potential epidemic? When biomedical researchers want to model potential treatment strategies on a richer dataset? When an employer is considering you for a job?

At present, policy and laws about online privacy and rights to information are challenging to interpret and difficult to enforce. As IoT technologies become more pervasive, personal information will become more valuable to a diverse set of actors that include organizations, individuals, and autonomous systems with the capacity to make decisions about you.

Some have suggested that individuals should have a basic right to opt out, delete, or mask their information from systems in the IoT, providing one tenet of a potential IoT “Bill of Rights.” However, it may be infeasible or impossible for an individual to control all the data generated about them by IoT systems.

Interestingly, strong individual privacy rights may also mean less social benefit. Too many “opt-outs” may erode the public and private value of IoT datasets,³ negatively impacting their social benefit—imagine a Google map where locations come and go. The complexity of providing useful services subject to dynamic participation and evolving individual preferences may be extraordinarily complex to develop and administer.

2. Who is accountable for decisions made by autonomous systems?

As autonomous systems replace some human activities, we face the challenge of when and how these systems should be deployed, and who is responsible and accountable for their behavior. When your “smart” system

fails, is hacked, or acts with negative or unintended consequences, who is accountable, how, and to whom?

A high-profile example of this is autonomous vehicles, which make many decisions without “a human in the loop.” We currently expect automobile companies to be accountable if automotive systems, such as anti-lock brakes, fail. As cars begin to drive themselves, who should be responsible for accidents? As systems take on more decisions previously made by humans, it will be increasingly challenging to create a framework for responsibility and accountability.

3. How do we promote the ethical use of IoT technologies? Technologies have no ethics. Many systems can be used for both good and ill: Video surveillance may be tremendously helpful in allowing senior citizens stay in their homes longer and parents to monitor their newborns; they can also expose private behavior to unscrupulous viewers and unwanted intrusion.

In his highly popular and visionary books, Isaac Asimov posited four laws of robotics^{1,2} on the basic theme that robots may not harm humans (or humanity), or, by inaction, allow humans (humanity) to come to harm. Asimov's Laws provide a glimpse into the social and ethical challenges that will need to be addressed in the IoT. How do we promote and enforce ethical behavior by both humans and intelligent systems? Will we need to develop and incorporate “artificial ethics” into automated systems to help them respond in environments when there are good and bad choices? If so, whose ethics should be applied?

Toward a Framework for Thinking About Principles and Policy for the IoT

What might a general IoT governance model look like? In 2008, the Forum for a New World Governance developed the “World Governance Index” (WGI) focusing on peace and security, democracy and the rule of law, human rights, development and participation, and sustainability. These areas provide a roadmap for considering IoT governance. Mapping the WGI areas to the IoT indicates that we will need:

► *Policy for IoT safety, security and privacy*, requiring the development of


viable approaches promoting individual rights, data security, and trust, as well as disincentives and penalties for inappropriate behavior, corruption, and crime.

► *A legal framework for determining appropriate behavior* of autonomous IoT entities, responsible and accountable parties for that behavior, and determination of who can enforce compliance, how, and on what grounds.

► *Focus on human rights and ethical behavior* in the IoT, including a sense of how these would be enforced. This gets to the heart of the need for the IoT to promote human well-being and contribute to the advancement of society.

► *Sustainable development of the IoT* as part of a larger societal and technological ecosystem, including its impact on biological systems (for example, 3D-printed organs, implants), environmental systems, and natural resources).

We need to lay the groundwork now. The IoT should advance society and not just technology. The first step is to pursue the discussions, studies, task forces, commissions, and pilots that will help develop governance for an empowering and enabling IoT. Developing policy and legislation in newsworthy and opportunistic areas (for example, transportation) is essential, but not enough. We need to be thinking deeply *now* about broad IoT use and deployment, and how it can help create a more enlightened and civilized society. If we wait too long, we do so at our own risk.

Acknowledgment: Many thanks to Danny Goroff, Jim Kurose, Theresa Bourgeois, and colleagues at Google for insightful comments on drafts of this column. 

References

1. Asimov, I. *Robots and Empire*. Doubleday, 1985.
2. Asimov, I. “Runaround” in *I, Robot*. Gnome Press, 1950.
3. Goodman, E. *The Atomic Age of Data: Policy for the Internet of Things*. Aspen Institute Report, 2015.

Francine Berman is the Edward P. Hamilton Distinguished Professor in Computer Science at Rensselaer Polytechnic Institute, Troy, NY. She is an ACM Fellow.

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

For more on the future of the Internet of Things, see the “Inside Risks” column, p. 26.

Copyright held by authors.



Association for
Computing Machinery

ACM Conference Proceedings Now Available via Print-on-Demand!

Did you know that you can now order many popular ACM conference proceedings via print-on-demand?

Institutions, libraries and individuals can choose from more than 100 titles on a continually updated list through Amazon, Barnes & Noble, Baker & Taylor, Ingram and NACSCORP: CHI, KDD, Multimedia, SIGIR, SIGCOMM, SIGCSE, SIGMOD/PODS, and many more.

For available titles and ordering info, visit:
libraries.acm.org/pod

Use the Scientific Method in Computer Science

IN “SEX AS AN ALGORITHM: The Theory of Evolution Under the Lens of Computation” (Nov. 2016), Adi Livnat and Christos Papadimitriou argued eloquently that the extraordinary success of sexual evolution has not been adequately explained. Somewhat paradoxically, they concluded that sex is not particularly well suited to the task of generating “outstanding individuals.” They also said that genetic algorithms are similarly ill suited to this task.

It should be noted that this critique of genetic algorithms—widely used derivative free optimization heuristics modeled on recombinative evolution—stands in counterpoint to a voluminous empirical record of practical successes. It also speaks to the long-standing absence of consensus among evolutionary computation theorists regarding the abstract workings of genetic algorithms and the general conditions under which genetic algorithms outperform local search. A consensus on these matters promises to shed light on the question the authors originally aimed to answer: Why does recombinative evolution generate populations with outstanding individuals?

Generative hypomixability elimination¹ is a recent theory that addresses this question, positing that genetic algorithms efficiently implement a decimation heuristic that generates fitter populations over time by iteratively eliminating the joint entropy of small collections of “hypomixable loci,” or loci in which alleles do not mix well. Recombination, or mixing, allows such loci to go to fixation even as it safeguards the marginal entropy of non-interacting loci.

Taking a step back, one might ask how this theory and the theory proposed by Livnat and Papadimitriou might be evaluated. Proof of soundness, wherever possible, is always desirable, but end-to-end proof can be elusive when analyzing computation in biological systems like brains and evolving populations. We must

instead use the scientific method,² an approach undergirded by the following rule:

$$\begin{aligned} \text{hypothesis} \implies \text{prediction} &\equiv \\ \neg \text{prediction} \implies \neg \text{hypothesis} & \end{aligned}$$

Unlike the foundations of, say, physics, the foundations of computer science are logically verifiable; hypotheses play no part. So, while computer scientists have seen engineering revolutions aplenty, they have seen nothing like the transition from a Newtonian universe to an Einsteinian universe or from the phlogiston theory of combustion to Lavoisier’s oxygen-based theory or any of the other foundational shifts described in Thomas Kuhn’s *Structure of Scientific Revolutions*. Theoretical physicists, chemists, and biologists trained informally, if not formally, in the application of the scientific method know how to evaluate and work with competing hypotheses. The same cannot be said of theoretical computer scientists today. For them, the scientific method is unfamiliar terrain, with different rules and alternate notions of rigor. For example, assumptions must be weak, and hypotheses testable.

For all computer science as a field has to contribute to the natural sciences, it also has much to learn.

References

1. Burjorjee, K.M. Hypomixability elimination in evolutionary systems. In *Proceedings of the 13th Foundations of Genetic Algorithms Conference* (Aberystwyth, U.K., Jan. 17–20). ACM Press, New York, 2015, 163–175.
2. Popper, K. *The Logic of Scientific Discovery*. Routledge, London, U.K., 2007.

Keki M. Burjorjee, Berkeley, CA

While Adi Livnat and Christos Papadimitriou’s article (Nov. 2016) provided the rationale for a provocative magazine cover, the article itself began with a false claim and ignored a much simpler explanation for the success of sexual evolution. Shortly after life appeared on Earth, approximately 3.8 billion years ago, evolution began diversifying lifeforms in a very pragmatic way, with mutations that increased

the ability of individuals to survive and reproduce being passed along to future generations, whereas those that were disadvantageous were naturally dropped. This process soon discovered that sexual reproduction worked better than simply subdividing, in that it allows advantageous mutations that occur in different families to be combined, allowing evolution to proceed more rapidly, whereas subdividing does not allow it. Sexual reproduction thus became dominant.

Nevertheless, the article said, “What is the role of sex in evolution? Reproduction with recombination is almost ubiquitous in life (even bacteria exchange genetic material), while obligate asexual species appear to be rare evolutionary dead ends. Yet there is no agreement among the experts as to what makes sex so advantageous.”

How can there be no agreement when the reason for sexual evolution is so obvious? In order for sexual evolution to work, each generation must die, which some people view as inconvenient, prompting them to imagine an afterlife. Subdividing, on the other hand, produces potential immortals who are naturally less diverse because they mutate less radically than the sexy species.

P.S. I do not hold any of this against Christos Papadimitriou, who I have known for 50 years.

Lester Earnest, Stanford, CA

Authors Respond:

Earnest’s idea, first proposed by R.A. Fisher (1930) and H.J. Muller (1932), does not solve the problem and is referenced in our online appendix where the interested reader can begin to explore this fascinating topic. The debate among experts is ongoing, and our recent article contributed a fresh idea to it. Burjorjee did not back up with evidence his claim of empirical success of genetic algorithms, compared to, say, simulated annealing. And a propos philosophy of science, he may refer to Papadimitriou’s 1995 article “Database Metatheory: Asking the Big Queries” (<http://dl.acm.org/citation.cfm?id=211547>) with its sections on T. Kuhn,

K.R. Popper, and P. Feyerabend, and their relevance to computer science.

Adi Livnat, Haifa, Israel, and
Christos Papadimitriou,
Berkeley, CA

Dismayed by ‘Sex’ Cover

I am writing to express my dismay and disappointment at the cover of the November 2016 issue introducing the article “Sex as an Algorithm: The Theory of Evolution Under the Lens of Computation” by Adi Livnat and Christos Papadimitriou, finding it offensive and attention-grabbing in a way that is inconsistent with ACM’s public mission.

While I would guess that most readers either do not care or thought the cover “funny” or “cute,” I have talked to enough of my colleagues, who describe their reaction as “shocked,” “appalled,” “offended,” and “embarrassed,” to believe it is a serious issue that warrants further reflection.

Specifically, is it really appropriate for ACM, a professional organization that purports to represent and support all its members and all members of the computing discipline, to distribute an issue that some are embarrassed to receive in our mailbox, display on our desks or conference tables, or look at on our computers if somebody might be looking over our shoulders?

First, the research in question is not about sex but about sexual reproduction and its effect on diversity in populations. There is a major difference, and conflating the two in this way comes across as juvenile. I cannot help think of “locker room talk.”

Second, placing the huge, bold-faced word “Sex” on a hot pink cover creates an obvious and immediate association with women. Given the underrepresentation of women in the field, this kind of message is completely counterproductive and particularly reminds young women, who may be less certain about how welcome they are in the field, that they are to be associated with sex, not science.

Third, the unfortunate timing of this issue, which arrived during National Breast Cancer Awareness Month, was undoubtedly unintentional, but to those of us who have lost loved ones to breast cancer, the hot pink cover felt disrespectful and insensitive.

This may not seem like a big deal, and I am sure some readers are thinking I am overly sensitive and humorless. But quite honestly, it is tough enough being a woman in an extremely male-dominated field without feeling embarrassed and awkward about displaying my own professional organization’s magazine in public.

In the end, I dropped it into the recycling bin without reading it.

Marie desJardins, Baltimore, MD

Editor-in-Chief Responds:

The cover in question, for which I am ultimately responsible, was meant to be humorous. Since several readers were offended by it, it is clear in retrospect the humor was misguided. For that, I sincerely apologize. This has been discussed by the design team, and we hope to learn from this mistake.

Moshe Y. Vardi, Editor-in-Chief

No Revolution Yet for Blockchain

Sarah Underwood’s news article “Blockchain Beyond Bitcoin” (Nov. 2016) was yet another disappointing read on blockchain, offering an (incomplete) summary of publicly available information on the technology and its proposed application areas. Many claims, including the key one that “Blockchain technology has the potential to revolutionize applications and redefine the digital economy,” were neither discussed nor backed up with evidence. From a scientific point of view, this is insufficient. Worse, like many blockchain proponents, Underwood failed, in my opinion, to raise the right questions. Instead of focusing on “what blockchain could do,” one should address “what blockchain can do better than other technologies.”

In this context, blockchain is often compared to existing solutions rather than to existing technologies, as in the proverbial comparison of apples and oranges. There may be any number of reasons, including operational, economic, or social, why an existing solution (as inadequate as it may be) has not been replaced in the marketplace. However, this does not mean per se there is no existing, better-understood technology than blockchain available to address a given problem.

Moreover, blockchain is often credited with the ability to solve tough long-standing problems. For example, Underwood mentioned “digital identity.” Various attempts to address this challenge, including well-established approaches (such as Public Key Infrastructure and Web of Trust) fail in various ways due to nontechnical aspects of human relationships, including trust, social, cognitive, economic, and even physical. So far, moreover, no evidence has been produced that shows how blockchain outperforms existing technologies in addressing the problem of digital identity.

It is time to ask the right questions about blockchain if we want to understand its actual properties, strengths, and weaknesses, as well as its promise.

Ingo Mueller, Melbourne, Australia

Keep Human Judgment in Remote Warfare

Though Keith Kirkpatrick’s news article “Can We Trust Autonomous Weapons?” (Dec. 2016) was thoughtful and well balanced, we must still ask how we should be identifying targets. From the prisoners in the U.S. military prison at Guantánamo Bay, Cuba, to the fighters targeted by autonomous vehicles in Pakistan or Yemen, we depend on human intelligence on the ground to choose the ones to target, even though that intelligence is sometimes faulty or false. In practical terms, we have shown we cannot get our boots *off* the ground. We need to embed our forces in and work with the populations we wish to protect. Remote warfare—unless separated completely from ethics, responsibility, and long-term consequences—is likely to remain a fantasy. It thus raises the perennial question of where to draw the line between computing intelligence and human reason, as explored by MIT professor Joseph Weizenbaum in his classic 1976 book *Computer Power and Human Reason: From Judgment to Calculation* on automation and human decision making. Where computerized warfare is concerned, human judgment remains the supreme arbiter.

Andy Oram, Boston, MA

Communications welcomes your opinion. To submit a Letter to the Editor, please limit yourself to 500 words or less, and send to letters@cacm.acm.org.

©2017 ACM 0001-0782/17/02

Artificial Intelligence

SINCE ITS INAUGURATION in 1966, the ACM A.M. Turing Award has recognized major contributions of lasting importance to computing. Through the years, it has become the most prestigious award in computing. To help celebrate 50 years of the ACM Turing Award and the visionaries who have received it, ACM has launched a campaign called “Panels in Print,” which takes the form of a collection of responses from Turing laureates, ACM award recipients and other ACM experts on a given topic or trend.

ACM’s celebration of 50 years of the ACM Turing Award will culminate with a conference June 23–24, 2017 at the Westin St. Francis in San Francisco to highlight the significant impact of the contributions of ACM Turing laureates on computing and society, to look ahead to the future of technology and innovation, and to help inspire the next generation of computer scientists to invent and dream.

For the first Panel in Print, we invited 1994 ACM Turing laureate **RAJ REDDY**, 2012 ACM Prize in Computing recipient **JEFF DEAN**, 2013 ACM Prize in Computing recipient **DAVID BLEI** and 2013 ACM Grace Murray Hopper recipient **PEDRO FELZENSZWALB** to respond to several questions about Artificial Intelligence.

What have been the biggest breakthroughs in AI in recent years and

what impact is it having in the real-world?

RAJ REDDY: Ten years ago, I would have said it wouldn’t be possible, in my lifetime, to recognize unrehearsed spontaneous speech from an open population but that’s exactly what Siri, Cortana and Alexa do. The same is happening with vision and robotics. We are by no means at the end of the activity in these areas, but we have enough working examples that society can benefit from these breakthroughs.

JEFF DEAN: The biggest breakthrough in the last five or so years has been the use of deep learning, a particular kind of machine learning that uses neural networks. Stacking the network into many layers that learn increasingly abstract patterns as you go up the layers seems to be a fundamentally powerful idea, and it’s been very successful in a surprisingly wide variety of applications—from speech recognition, to image recognition, to language understanding. What’s interesting is we don’t seem to be near the limit of what deep learning can do; we’ll likely see many more powerful uses of it in the coming years.

PEDRO FELZENSZWALB: Among the biggest technical advances I would include the development of scalable machine learning algorithms and the computational infrastructure to process and interact with huge datasets. The latest example of these ad-

vances is deep learning. In computer vision deep learning has led to breakthroughs in object recognition. The accuracy of object recognition in popular benchmarks has increased way beyond what most of us expected to see in the last few years. The impact of this progress still remains to be seen but I expect it will play an important role in building intelligent systems that can interact directly with our physical world.

What specific AI applications will most improve our quality of life in the next five years, 10 years?

JEFF DEAN: Three areas stand out for me: healthcare, self-driving cars, and general-purpose robotics. Machine learning systems will be able to offer suggestions and advice to doctors in ways that are very complementary to the strengths of human medical professionals, resulting in better care for patients, and more efficient healthcare systems. Self-driving cars will be incredibly transformative as well: our urban environments are built around the idea that people own cars and need to park them, etc., and we’ll start to see dramatic changes in even things like how cities and neighborhoods are designed as self-driving cars become more widespread. General purpose robots that can operate in messy, uncontrolled environments like households or offices will also start to have a big impact in this time frame.

DAVID BLEI: I believe that we are now



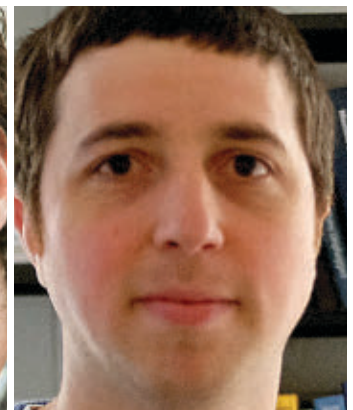
1994 ACM Turing laureate
Raj Reddy



2012 ACM Prize in Computing
recipient Jeff Dean



2013 ACM Prize in Computing
recipient David Blei



2013 ACM Grace Murray Hopper
recipient Pedro Felzenszwalb

making major progress in two areas that will significantly improve our quality of life. The first is in natural language processing, both in language understanding and language generation. The second is in personalization, in developing software and methods that adapt to user behavior.

These two threads of innovation will result in a more seamless interface between people and AI software, enabling AI to help our lives and society in more ways. For example, we will be able to carry on intelligent and useful conversations with an algorithm, especially around question answering of existing facts. The seamless interface—powered by natural language understanding and personalization—will change how we interact with knowledge bases such as libraries and the internet and thus change how we are able to access, find, and use information.

PEDRO FELZENSZWALB: I believe medicine and public health are areas where the potential for AI is very big and we may see significant impact in the next 10 years. Consider the problem of medical diagnosis. Conceptually this is a simple problem, involving figuring out what condition someone has based on their symptoms. But in practice the problem is very hard.

We rely on specialists, as no one doctor can master all the complexities of the human body. An AI doctor will have access to a database with all of our medical knowledge and the necessary computational capabilities to reason about this data. This AI doctor could be much more easily accessible than the best doctors in the world. The bottom line is that medical diagnosing requires doing statistical inference with lots of data, something that computers can probably do better than humans.

What are some of the major hurdles that AI still needs to overcome in the next 10 years?

DAVID BLEI: Right now, AI is revolutionizing technology through prediction, e.g., “What will I buy next?” or “What face is this in the picture?” I believe that AI will next revolutionize science and scholarship, i.e., how we understand our world through observation. In the context of many fields—astronomy, genetics, sociology, his-

“In my opinion, we are still quite far from realizing the potential of AI. One meta-hurdle is to define what we mean by intelligence.”

tory, and many others—AI can help us analyze massive collections of data to form an understanding of what happened and how things work.

But there is a significant hurdle to this vision. Finding causal connections, e.g., for science and history, is a deep statistical problem. We must develop the field of causal inference in the context of modern AI to realize its potential in this way.

I will add that using AI to find causal connections will also have an impact technologically. Problems around medical personalization—such as how will a particular patient respond to a medicine—might seem “predictive” at first, but are ultimately causal questions. Indeed, using AI for causal inference will only bolster our predictive capabilities.

PEDRO FELZENSZWALB: In my opinion, we are still quite far from realizing the potential of AI. One meta-hurdle is to define what we mean by intelligence. In the history of AI we have had some specific goals, such as building a computer that can play chess as well as any human, or getting a computer to recognize objects in pictures. However, the AI community has often looked down upon practical solutions to such problems, citing among other things that large engineering efforts and special purpose solutions have little resemblance to intelligence and will not generalize to other problems. It appears that as soon as we figure out how to solve a classical problem in AI we no longer consider the problem to be part of AI. Perhaps the solution simply demystifies the problem too much. It is not clear if we will ever at-

tribute intelligence to a system that we fully understand.

Much has been made of the potential for AI in pop culture. What are some of the biggest myths you’ve seen? Can you think of examples where science fiction is getting close to reality?

JEFF DEAN: Probably the biggest myth is that AI is one singular thing that you can just “flip on” like a switch, and suddenly you’ve got human-style intelligence. In fact, AI is a huge field involving many techniques, only very loosely inspired by human intelligence. The good news is these techniques are already quite practical for some kinds of real-world applications today—this is why you can talk to Google on your phone, and it understands what you mean and can give you good answers. It’s not magic, but it already works well enough that it’s really impressive compared to what we could do just a few years ago.

RAJ REDDY: The best example is Ray Kurzweil and Vernor Vinge’s description of the singularity which I believe will happen. Where we disagree is on “when” it will happen. I think it won’t happen for at least another 100 years, if not longer.

Two of my favorite examples of science fiction in the movies are *Minority Report* and *Her*, not because they are completely realistic, but because they provide a plausible scenario of things that could happen. In my Turing talk, I speak about teleportation, time travel, and immortality, but then I go on to redefine what I mean by those terms. For example, if we can observe things happening in 3D Virtual Reality without physically being there, that, in my mind, is teleportation, but of course that’s not the same definition you get from things like *Star Trek*. The same thing happens in mathematics. If mathematicians don’t like a particular outcome, they will define a new complex number world where such facts tend to be true. The issue is, if you don’t like the world that you are in, then make a world where what you are imagining is true. There are lots of possibilities, some are reasonable and others may not be, but that depends on the date and time when you ask the question. ■

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:10.1145/3022177

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

Liberal Arts Academia Wants YOU!

Janet Davis makes a plea to CS practitioners to consider even a short teaching stint.



Janet Davis
**Tech Industry Ph.D.'s:
Academia Can Be
Nicer Than You Think**
<http://bit.ly/2efaNHY>
October 21, 2016

Dear industry colleagues:

I understand some of the reasons why you took your Ph.D. and ran: the glamour and mystery of Silicon Valley. The opportunity to work with smart colleagues on products that make a difference in people's lives. The excitement of a rapidly changing field. The opportunities to change jobs and take sabbaticals. The hours. The perks. The paycheck.

And, being in industry means never having to write a grant proposal.

I, too, recoiled at the thought of a research career in which I would spend my time writing grant proposals, managing grad students, and writing more proposals so I can continue to support my grad students, never touching a line of code. I, too, considered a career in industry, albeit briefly. Instead, I've pursued a career teaching computer science (CS) at liberal arts colleges, along with a little research and more administration than I ever thought I would. Ten years and still going strong!

Liberal arts colleges differ from research universities in some key ways:

- ▶ Liberal arts colleges focus on undergraduate education. A liberal arts college does not grant Ph.D.'s. Some offer masters' or professional degrees, but these programs tend to be quite small.

- ▶ Liberal arts students learn many different ways of thinking. In a typical engineering program, major requirements might constitute 60% of a student's coursework, or even more. At a liberal arts college, major requirements typically comprise 25%–35% of a student's coursework, with the remainder spent exploring other disciplines, and perhaps developing depth in a second discipline through a double major or minor.

- ▶ To earn tenure, liberal arts faculty are typically expected to demonstrate excellence in teaching, an active program of scholarship, and contributions to their institution and profession. Faculty might spend 60% of their time on teaching, 30% on scholarship, and 10% on service.

Teaching at a liberal arts college has more in common with working in the tech industry than you might think:

- ▶ **The glamour.** Silicon Valley is glamorous, but liberal arts profs are

mini-celebrities among their students and often in the towns where they work. "Scientist" and "teacher" are the fourth and fifth most-admired professions in the U.S. (<http://bit.ly/2gcVrzb>). Chances are good you'll be the *only* local expert on your research area, and many other things besides. It's a small pond, but we are all big fish!

- ▶ **The mystery.** Most people don't understand our jobs any more than they understand yours.

- ▶ **Working with smart people.** Most academics are pretty smart. Working at a liberal arts college means nearly all of your colleagues are *not* computer scientists, which means you can learn a lot just by talking with them.

- ▶ **Meetings.** We have meetings just like you do. What a great opportunity to learn from our diverse and thoughtful colleagues! I've gotten to know some really cool people by serving on committees with them.

- ▶ **Making things.** Teaching is all about creating learning experiences. I still write code: I write my students' homework assignments.

- ▶ **Getting your hands dirty.** Undergrad researchers need supervision. Chances are good you will find yourself working beside them, whether to teach them how to do it right or figure out what they did wrong. You'll get to investigate bugs you never even dreamed possible!

- ▶ **Making a difference.** I admit it: nobody uses the things we academics make. *Except our students.* Strangely enough, students mostly do what we tell them to do; they seem to trust our in-

structions are intended for their learning. Sometimes, students and alumni come back and tell us how much they learned from our classes, projects, and assignments. Not to get sentimental, but in the last month I've gotten notes from two alumni thanking me just for spending time with them.

► **The excitement of a rapidly changing field.** It seems I never teach the same course twice. What we teach in academia is influenced by what you are doing in industry, so we're changing along with you, and the body of foundational knowledge in CS is still expanding. For example, between studying computer architecture as an undergrad in 1996 and teaching it for the first time in 2011, pipelining and superscalar architectures made it into the textbooks. No big deal, right? CS is also expanding into interdisciplinary applications. Liberal arts colleges value learning across disciplines, and our relatively flat organizational structures facilitate collaboration. Even if the technology and content of a course haven't changed, the *students* have.

► **Changing jobs.** It seems like folks in Silicon Valley change jobs every 2–3 years. I change jobs *all the time*. Every semester I am teaching different courses on a different schedule. Every year I have different administrative responsibilities. Bored? Frustrated? Wait three months!

► **Sabbaticals.** Sabbaticals are a unique opportunity in the tech industry, compared to most other industries. (As fast as everyone changes jobs, no one will notice a few months' gap on your résumé.) By contrast, sabbaticals are *de rigueur* for faculty. The traditional sabbatical is one year out of every seven, but some colleges offer more than that. The truth is, we need it—time to renew, reflect, research, write, and *rest*. Undergraduates are young. We're old. They only stay for four years. We'd die if we tried to keep up with them all the time.

► **Summers.** You don't get summers off? We do! Well, not exactly off; most liberal arts college faculty have nine-month contracts, so we are on our own over the summer. Some use this time to take on short-term contract work, or write summer salary into their (gasp!) grant proposals. Most write, research, travel, supervise students, and/or prepare for the coming academic year. It's important to rest during our summers,

too, even if we sometimes feel guilty for not getting as much done as we planned. Fortunately, our nine-month salaries are spread out over all 12 months of the year, so we don't starve.

► **The hours.** Like you, we sometimes work long hours to meet release deadlines. That means not just meeting conference deadlines, but also getting big assignments out, finalizing grades, or making recommendations for hiring. Much of our work falls into a routine, and we don't get brownie points for arriving early or staying late. No one is watching (except our students, who have been known to remind the less self-disciplined of us to go home and eat dinner). Much of our work is our own choice—usually how we do it, and often what we do. When I've had the longest hours, I've been doing things I enjoy: meeting interesting people, teaching new classes, and working with my students. Also, see the previous “Summers.”

► **The perks.** You know all those perks that Google and Facebook offer? They are trying to create a college-like environment. Why not go for the original? Free lunch might not be offered every day, but it's often an incentive for attending seminars and committee meetings. The dining hall is cheap. Candidates and speakers need to be taken out to dinner at fancy restaurants. Why is there so much alcohol at faculty events? To get us to stop working on our own things and talk to each other. Many colleges offer inexpensive housing near campus for new faculty, so you can roll out of bed and walk to class just like your students do (though maybe not in your pajamas). Trust me, after a couple of years, you won't want to be quite so close to campus.

► **The paycheck.** I won't lie: academic paychecks are nowhere close to Silicon Valley paychecks. Though computer scientists tend to earn more than most faculty, some of your new grads will earn a higher salary than you do. There are no annual bonuses or stock options in academia. On the other hand, the same salary goes a lot farther in Grinnell, IA, or Walla Walla, WA, than in Silicon Valley, and in the eternal words of Jessie J: “It's not about the money.”

► **Funding research.** While most liberal arts colleges will help you write external grant proposals—and I have many colleagues who have received ex-

ternal research grants—few liberal arts colleges require faculty to seek external grants. If your equipment needs are modest, there is often internal funding to support students. Internal proposals are reviewed not by other computer scientists (or by the C suite), but by colleagues in the liberal arts. This means as long as you can explain it convincingly and eventually get it published, you can do pretty much whatever kind of research you want. There's no need to reframe your real interests to fit a corporate or national research program.

I haven't written this just to hear myself talking (though it sometimes seems faculty are prone to that). CS is facing a crisis in hiring. An increasing number of new Ph.D.'s are bound for industry, which means faculty positions are going unfilled. Whatever good experiences you had as a CS undergrad, students today are not necessarily getting the same experience. Fewer faculty means bigger classes, less hands-on mentoring, and a chillier climate for women and minorities. We need you to help us make undergraduate education everything it should be.

Thinking long term, if more Ph.D.'s don't return to the academy, there won't be enough graduates capable of taking all those high tech jobs. Thinking even longer term, if there are not enough CS graduates getting Ph.D.'s, the whole pipeline could grind to a halt (<http://stanford.io/2fQl8EL>). Liberal arts colleges make a real contribution to maintaining the Ph.D. pipeline: in measuring yield of science students who go on to earn Ph.D.'s (<http://bit.ly/2fKvUkL>), baccalaureate colleges are second only to “research universities with very high research activity,” and constitute the majority of the list of top 50 institutions. It is critical we get new faculty who can fill open positions at liberal arts colleges.

Please consider a return to academia, whether forever or just for a year. I promise, there will still be jobs in Silicon Valley when you go back.

Signed,

Your undergrad professors
(We always believed in you!)



Janet Davis blogs on her experiences as Whitman College's founding computer scientist at <http://blogs.whitman.edu/countingfromzero/>.

© 2017 ACM 0001-0782/17/2 \$15.00



Seeking applications from outstanding young leaders

The ACM Future of Computing Academy will bring together next-generation researchers, practitioners, educators and entrepreneurs from various disciplines of computing to define and launch new initiatives that will carry us into the future. Academy members will have the satisfaction of contributing to our field while enjoying the opportunity to grow their personal networks across regions, computing disciplines and computing professions. ACM invites accomplished professionals typically in their 20s and 30s to apply.

The inaugural meeting of the ACM Future of Computing Academy will be June 25, 2017 in San Francisco. Members of the Academy will be invited to attend ACM's celebration of 50 Years of the ACM Turing Award, June 23-24, at the Westin St. Francis, where they will have the opportunity to interact with ACM A.M. Turing Award laureates.



"Academy members will have the privilege and responsibility of being the voice of the next generation of computing professionals and ensuring that ACM continues to contribute to their success long into the future."

– **Vicki Hanson**, ACM President

"The Future of Computing Academy will give some of the most talented, creative, and passionate young computing professionals a collective voice that will help shape the future of our industry and its influence on our social and economic ecosystem."

– **Vint Cerf**, Google Chief Internet Evangelist and former ACM President



"The Future of Computing Academy will afford members an invaluable opportunity to expand their professional networks to include outstanding individuals with demonstrated excellence from across a breadth of computing disciplines."

– **Aaron Quigley**, Chair of Human Computer Interaction, University of St. Andrews

"Members of the Academy will have opportunities to interact with computing pioneers whose foundational contributions influence innovation today."

– **Matthias Kaiserswerth**, Managing Director, Hasler Foundation



Apply at: <http://www.acm.org/fca>



Association for
Computing Machinery

Secure Quantum Communications

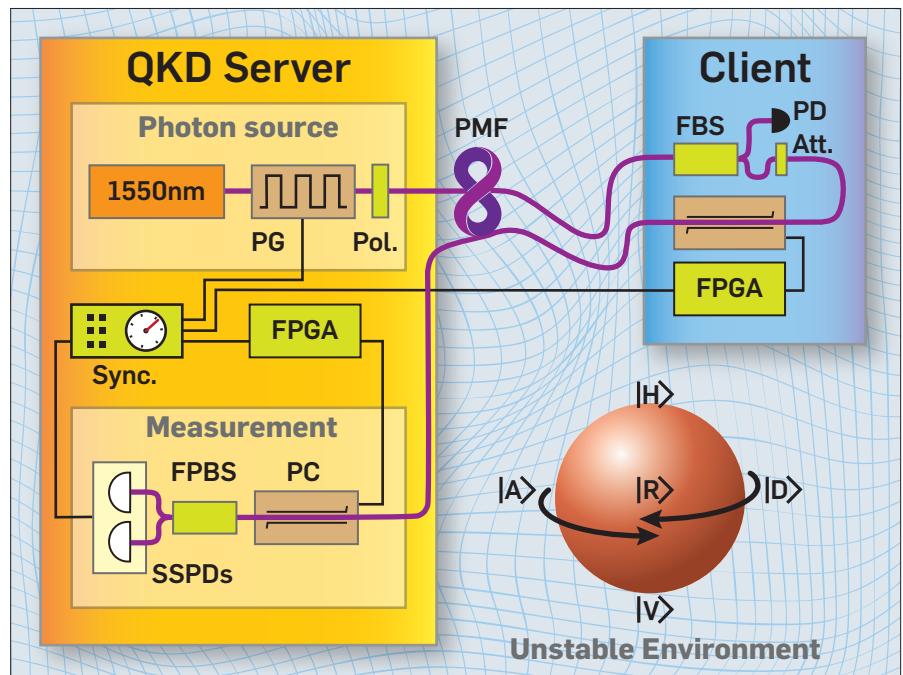
Data locking experiments provide stepping stones to a possible future in quantum cryptography.

TWO INDEPENDENT EXPERIMENTS published on the same day last August (August 12, 2016) have demonstrated the potential for quantum mechanics to improve the efficiency of secure data communications.

The experiments mark a departure from the current mainstream proposal for quantum communications: quantum key distribution (QKD). First demonstrated in 1989, QKD was designed to provide classical communications channels with a more secure method for delivering keys through the use of quantum mechanics.

“QKD uses a quantum channel that is able to transmit quantum states to establish a secure link between a sender and an intended receiver. But the information is actually transmitted over a classical channel, such as a telephone line,” says Daniel Lum, post-graduate researcher at the University of Rochester, and the lead author on the paper in *Physical Review A* that describes one of the two new experiments on what its proponents call quantum data locking (QDL).

Yang Liu, lead author of the other QDL paper in *Physical Review A* and a researcher at the Shanghai branch of



Experimental set-up for client-server reference frame independent quantum key distribution (rfiQKD). Source: P. Zhang, et al. arXiv:1308.3436 [quant-ph]

the Hefei National Laboratory for Physical Sciences at the Microscale of the University of Science and Technology of China, adds: “QKD is a mature technology today, both from a theoretical and from a practical view. The implementation and performance of QKD is

good enough for commercial products. It is an important primitive for encryption. The QDL protocol, on the other hand, is relatively new.”

David DiVincenzo and colleagues from IBM Watson Research Center and the University of Gdansk proposed the

basis for QDL in 2004. Although it has attracted far less academic and industrial attention than QKD, it has developed over the past decade into a family of theoretical encryption techniques.

“QDL shows a phenomenon that is unique to quantum information theory; it is not possible in classical information theory. In application, we have demonstrated the QDL protocol is able to encrypt a message and send it over tens of kilometers of fiber,” Liu claims.

Although it is much younger and faces a number of challenges, the attraction of QDL over QKD is that it is potentially far more efficient in terms of how much information can be encrypted for each bit of key than any system that relies on classical communication.

To provide provably secure communication, protocols in use today need to obey a theory developed by Claude Shannon in the 1940s. The encryption key, which must be generated randomly, needs to be the same length or greater than the information content of the message itself. Shannon’s theory provided support for the one-time pad developed in the late 19th century, in which sender and receiver agree to use a common key—originally taking the form of characters written on a pad of paper—only once. Once the message had been received and decoded, the key was to be discarded.

QKD provides the means for two parties, Alice and Bob, to agree on a secret key without risk of it being obtained by an eavesdropper. The protocol takes advantage of the way in which an attempt to determine one part of the quantum state of a particle disturbs the others. It makes it impossible to completely determine the quantum state of a photon or particle and, as a result, copy it.

Under QKD, when taking measurements of a sequence of photons they exchange, Alice and Bob agree to randomly swap between two different types of measurement of the quantum state and then compare the results. Eve can intercept the photo, perform her own measurements, and attempt to copy the photo and pass it on to Bob. They cannot, however, determine the state of the other property, and the new photon will be

forwarded with a state that probably does not match Alice’s original.

Without an eavesdropper like Eve present, Alice and Bob’s measurements will match approximately half the time because of their random switching between properties. With an eavesdropper present, the error rate rises significantly, because of the 50% probability for each photon that the eavesdropper has picked the wrong measurement to perform. But if enough of Alice’s and Bob’s measurements agree, the received pattern becomes a shared private key that can be used to encrypt messages on another channel, which can use traditional classical coding techniques.

One problem for QKD is the limit on communication speed caused by the nature of the protocol itself, combined with the effects of noise and interference in the quantum channel. Stefano Pirandola, a researcher at the University of York, says QKD protocols based on encoding pairs of properties into ‘qubits’ tend to deliver very low key-update rates. One way to boost the update rate is to use continuous-variable properties such as the quadrature operators of the coherent light transmissions from lasers. These quadrature operators “play the same role that position and momentum play for a particle such as an electron,” he says.

The need to use lengthy keys for message delivery still leaves QKD-based systems facing a potential bottleneck. QDL can harness the difficulties eavesdroppers have in intercepting quantum channels to send the data bits themselves and use exponentially shorter keys than those needed for Shannon’s one-time pad system.

To employ QDL, Alice and Bob first agree on a shared key, which could be

QKD makes it impossible to completely determine the state of a photon or particle and, as a result, copy it.

generated using QKD. That key selects a set of codewords that determine the sequence of properties to be measured and their contents. Each codeword calls for a different sequence of measurements on the quantum states. As with QKD, Eve can only access a fraction of the complete message, even with access to unlimited computing power.

“I think QDL is an interesting approach that relies on the realistic assumption that today, an eavesdropper cannot do everything and can only access quantum memories with limited lifetimes,” says Pirandola.

Liu explains, “We performed two experiments using our setup. The first was to show the original data-locking idea. The protocol locks half of the message using a 1-bit key. With a key length of one, the maximum information the eavesdropper may obtain is half of the message Alice sent.

“The second experiment was towards more practical schemes, limiting Eve’s information to an arbitrarily small amount using logarithm-length keys.”

Using free-space transmission rather than fiber allowed Lum’s team to explore higher dimensions of encoding based on more complex combinations of quantum properties to allow the transmission of error-correction bits along with the message encoded into the photon’s state. However, there is a trade-off inherent in the use of error correction; the redundancy it introduces makes it easier for an adversary to decrypt messages. As a result, a higher key ratio is needed to guarantee security and successful communication over noisy channels.

Liu and Lum both stress the experiments they performed were proof-of-concept demonstrations. Some of the theoretical requirements for QDL are not possible to realize in practice. For example, the experiments by both teams used the same technique as that used for QKD to generate pairs of photons. However, spontaneous parametric down-conversion is a random process that can create more than two daughter photons, with uncertain timing. Neither is desirable for QDL.

“Many reviewers pointed out that our experiment was not stringent enough to be considered truly secure; we cannot guarantee that we limited

“The QDL scheme is still in its infancy; it shows new physics and reveals possibilities for new applications. There is still more science to be performed.”

the accessible information of an eavesdropper to an arbitrarily small amount because of optical losses, efficiencies, and the inability to transmit one photon on demand,” Lum says.

“The main weakness of the QDL, I believe, is in quantum-channel losses. To reliably transmit messages via quantum states is a challenging problem and any unpredictable changes to the quantum states in transmission will corrupt the data.”

One obstacle to both QKD and QDL is the question of distance. Experiments have demonstrated the ability to transmit photons that retain entanglement over several hundred kilometers in free space, and 150km in fiber. “For long-range communication, preserving the quantum state over a long-range quantum channel is a formidable challenge; many believe it simply isn’t practical,” Lum concedes.

The choice of quantum encoding also will limit transmission distance: “Continuous-variable systems are limited to metropolitan distances because of technological issues, but they completely out-perform qubit-based protocols in terms of achievable rates for QKD,” says Pirandola.

Evidence for the practicality of long-distance quantum communications may come from experimental satellites launched this year. Researchers from the SpooQy Lab at the Center for Quantum Technologies in Singapore planned to launch a satellite payload into low Earth orbit to perform quantum-communication experiments in 2014, but the launcher exploded short-

ly after liftoff. The researchers aim to have a replacement in space in the autumn, but they will follow Chinese researchers who had their Quantum Experiments at Space Scale (QUESS) satellite successfully inserted into a solar-synchronous orbit 600km above sea level in August. The Chinese satellite will relay quantum transmissions over thousands of miles between ground stations in China and Europe.

As with much of the research into communications, because of the protocol’s relative maturity, the satellite projects will focus on QKD issues such as preserving entanglement over large separations. But if the experiments are successful, they should demonstrate that QDL and other quantum protocols that may be developed have a practical future in communications security.

“QDL is not going to outperform QKD anytime soon, which is why many experts in quantum cryptography do not regard the QDL demonstrations as high impact. We acknowledge this and present our experiment as a stepping stone to a possible future in quantum cryptography,” Lum says.

Liu adds, “The QDL scheme is still in its infancy; it shows new physics and it reveals possibilities for new applications. There is still more science to be performed.” **□**

Further Reading

Liu, Y., et al
Experimental Quantum Data Locking. *Physical Review A*, 94, 020301 (2016).
Preprint: <http://arxiv.org/abs/1605.04030>

Lum, D., et al
A Quantum Enigma Machine: Experimentally Demonstrating Quantum Data Locking. *Physical Review A*, 94, 022315 (2016) Preprint: <http://arxiv.org/abs/1605.06556>

Wilde, M.M.
Quantum Information Theory, Second Edition. Cambridge University Press (2016). Preprint: <http://arxiv.org/abs/1106.1445>

Cheng, C., Chandrasekara, R., Chuan, T.Y., and Ling, A.
Space-Qualified Nanosatellite Electronics Platform for Photon Pair Experiments. *IEEE/OSA Journal of Lightwave Technology*, Vol. 33, 4799 (2015)

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

© 2017 ACM 0001-0782/17/2 \$15.00

ACM Member News

APPLYING MATHEMATICS TO GENOME-SCALE DATA



“People who love math just love math,” observes Tandy Warnow, a professor in the departments of

computer science and bioengineering at the University of Illinois at Urbana-Champaign.

Warnow earned B.A. and Ph.D. degrees in mathematics from the University of California at Berkeley. She says a post-doctoral fellowship at the University of Southern California, working with statisticians in biology, was a turning point for her “in realizing that there were things that people cared about that weren’t just how pretty and how hard the math is; the relevance to an application is what mattered. At the time, it didn’t influence what I did, but it influenced what I thought.”

After a second post-doctoral year at Sandia National Laboratories, Warnow joined the Computer and Information Sciences Department at the University of Pennsylvania, where “my trajectory really changed, from being a pure mathematician working on discrete math problems in biology, to really caring about developing methods that will work well on data and can be used by biologists and linguists to give them highly accurate analyses.”

Warnow then spent 15 years as a computer science professor at the University of Texas at Austin before coming to the University of Illinois, where she is writing a book on computational phylogenetics, and researching how to use genome-scale data to explore the evolution of species.

“You have to figure out how to do something that is computationally feasible, will have statistical guarantees, and will perform well in practice,” Warnow says. “It requires thinking about it as a discrete mathematician, a statistician, and as a computer scientist; you can’t just come at it from one direction.”

—John Delaney

Are Computer Chips the New Security Threat?

Security researchers have identified a technique for installing a backdoor on computer chips, a security flaw that could profoundly change the computing industry.

THROUGHOUT THE HISTORY of computing, a common assumption has always been that microchips are generally secure; while software may be infected with malware or nefarious backdoors, hardware could be mostly trusted. As Milos Prvulovic, a professor at Georgia Institute of Technology's School of Computer Science puts it: "Most people, even among security researchers, have not questioned the integrity of hardware. We have all assumed that the hardware we use works exactly as specified and that it reads all instructions correctly."

Although researchers and security experts have been concerned about the possibility of a Trojan Horse or other type of hardware attack, the danger has remained in the theoretical realm.

Until now.

In May 2016, a team of researchers at the University of Michigan, including Todd Austin and Matthew Hicks, presented a paper showing exactly how to sabotage a microchip. The pair purposely built a backdoor into a chip and presented an academic paper at the IEEE Symposium on Privacy and Security documenting the method (it captured the conference's Best Paper award). The security flaw could allow a nation-state or other nefarious entity to grab and steal data. "The vulnerability creates concern because it's a method that could actually be used to do harm," says Austin, a professor and director of the university's Center for Future Architectures Research.

The discovery has sent a shock wave through the computing field. "This is the most demonically clever computer security attack I've seen in years. ... It's an attack that can be performed by someone who has access to the microchip fabrication facility, and it lets them insert a nearly undetectable back-



An employee checking components at Infineon Technologies AG microchip and sensor manufacturing facility in Germany.

door into the chips themselves," wrote Yonatan Zunger, head of infrastructure for the Google Assistant. And while the theoretical concept of embedding malware in hardware is not particularly new, the project "demonstrates just how feasible and devastating this method can be," says Abhi Shelat, associate professor of computer science at Northwestern University.

Risky Chips

Although it is incredibly difficult to spot security flaws in software, finding them in hardware can be exponentially more

complex. Austin refers to the challenge as finding the proverbial needle in a haystack. The reason is fairly simple, even if the technique he and Hicks used is not. Security researchers have historically focused virtually all their attention on the digital level of abstraction. "Defense tools rely on finding ones and zeros to identify malicious code," says Hicks, a lecturer at the University of Michigan. However, "An attack doesn't have to play by the digital rules—and there are currently no tools for detecting such an attack."

As a result, Austin and Hicks focused their attention on the analog

domain. “We began to explore this space because there are an infinite number of values between zero and one,” Hicks explains. Although it is entirely possible for security researchers to detect malicious hardware using an inspection-based technique—if it is large enough relative to the circuit to view or there is some visible effect on the power, performance, or temperature of the chip—their approach circumvents this approach. It also sneaks around a key protection: functional verification, essentially checking to see that the behavior of the chip matches its specifications before the design is sent to a foundry for fabrication. Using functional verification, “It’s possible to check for reliability problems and other types of errors,” Austin says.

Their method? After the design phase is complete and the microchip is ready to be fabricated, the saboteur drops a single engineered component into the overall structure. Since today’s microprocessors contain as many as a billion cells, this single cell is essentially indistinguishable from the rest of the components, even though it is secretly designed to act as a capacitor—temporarily storing electrical charges—rather than handling regular functions. Then, when a malicious script from a website or application triggers an obscure command, the capacitor grabs a tiny electric charge and stores it in its wires without affecting the chip’s power or performance characteristics.

Once the chip hits a predetermined threshold (typically after thousands or tens of thousands of events), the capacitor flips on a logical function and grabs control of the operating system. “The system avoids the triggers that provide a clue something is wrong,” Austin explains. What is more, “It’s highly unlikely that defenders or anyone testing the system will accidentally stumble onto the attack method.” Adds Hicks, “Detection would require a piece of logic that specifically looks for an arcane and extremely rare sequence of instructions. This essentially renders the detection processes useless.”

One thing the researchers honed in on during the project was using a basic version of a counter-based trigger. A simple way to engineer the attack would have been to increase the counter by one every time a certain set

“Detection would require a piece of logic that specifically looks for an arcane and extremely rare sequence of instructions.”

of criteria were met, such as turning on or off the computer and storing the value in a flip-flop state. However, this requires digital circuits, and accompanying logic that exposes the attack to testing or visual side-channel analysis inspection. Instead, using the analog domain, the capacitor continually adds the charge and increases voltage as if it were filling up a bucket. Because the voltage stays between zero and one, it is invisible as a digital value. When it finally hits the one level, the triggering mechanism takes place. But since the secret value is actually analog voltage in the capacitor, it remains stealthy.

The researchers tested the system under a wide range of environmental conditions—including temperatures ranging from -13 degrees to 212 degrees Fahrenheit—and the process worked consistently. “The behavior only exists in the analog domain. So, if you try to analyze the environment with digital tools, the analog behavior disappears; it no longer exists. This makes it appear that the activity doesn’t exist at all,” Austin explains.

Adds Hicks: “The attack method uses the oldest trick in computer security. If you want to go undetected, then get below the things that detect you.”

Deep Insecurities

At this point, it appears nobody has used this approach in the wild. As far as everyone knows, Austin and Hicks were the first to break hardware into layers to create an attack method. Nevertheless, the risks are very real. Today, a relatively small number of chip fabrication facilities exist worldwide and no one can rule out the possibil-

ity that a worker at a facility could use this method to plant spyware or other code. Says Shelat, “Although only a handful of organizations are able to fabricate ASICs today, the reality is that they are now used for handling critical tasks and infrastructure.”

To be sure, Shelat says there is a real-world risk. “There is evidence of Tailored Access Operation (TAO)-style attacks mounted by sophisticated organizations,” he says. Using this method, “Physical hardware that has been ordered by the victim is intercepted and implanted with Trojan hardware that allows remote access. A natural extension of such attacks would be to manufacture a batch of chips with custom backdoor access, and then inject these into a supply chain that is incorporated into a target population. This would allow an organization to wreak havoc on critical systems while making it nearly impossible to isolate the flaw.”

What is particularly frightening about this method, Prvulovic says, is that it takes full control of a computing device and it is undetectable until activity reaches a certain threshold. At this point, “anything and everything is potentially compromised.” Moreover, there is no known antidote for the threat, though Austin and Hicks suggest some possible methods in their academic paper.

The upside, Prvulovic adds, is that chip fabrication does not take place overnight; in fact, in many cases, it takes years to design a chip. And while it is possible that someone could add a component in a shorter time frame, “This isn’t something that is likely to appear any time soon; though if it did, we almost certainly wouldn’t know about it. There’s also risk for a semiconductor company; if this is detected, your company is most likely out of business.”

Not surprisingly, the research team’s efforts have been greeted with both praise and disdain. Of course, most in the computing sciences field have come to acknowledge the value of exposing vulnerabilities and support the project. “Overall, this is a very positive thing,” Prvulovic says. “It isn’t something that requires an Einstein-level genius to figure out; it’s something that, if you think about it and work on it, you

might eventually stumble onto this approach. That's what makes it so dangerous, and that's why it's good that this is now out in the open."

Shelat adds that he and others in the field are genuinely impressed by the methods Austin and Hicks used. "Their attack is clever because it uses both digital and analog techniques to implement a privilege escalation attack." In fact, Shelat is now involved in research aimed at developing verifiable hardware for a limited class of circuits. The end-game is to develop "advanced cryptographic protocols in order to design a chip that can *prove* in real time that it has performed the correct computation." However, he admits the gap between theory and reality remains formidable, and many of the brightest minds in computing have focused on this concept for decades.

Austin and Hicks say they have already briefed members of the U.S. Department of Defense and various branches of the U.S. military, as well as chip manufacturers and others about the attack method and how it could

be used. They also have given some of the 100 chips they fabricated to government officials and industry executives.

Says Hicks, "The key to addressing these risks is to not stick our heads in the sand, but rather encourage research on analog circuits and the risks associated with them when they are part of digital systems."

Adds Austin, "There are people who were very upset about this research, but if we all stick our head in the sand together, the threat will not go away. We are hoping that this research will spur more attention on analog circuits and the risks associated with them when they are part of digital systems." ■

Further Reading

Yang, K., Hicks, M., Dong, Q., Austin, T., and Sylvester, D.

A2: Analog Malicious Hardware. 2016 IEEE Symposium on Security and Privacy. July 2016. http://static1.l.sqspcdn.com/static/f/543048/26931843/1464016046717/A2_SP_2016.pdf?token=QXoVmAnDwRuiL84oo13X0iH6cXI%3D

Wahby, R.S., Howald, M., Garg, S., Shelat, A., and Walfish, M.
Verifiable ASICs, *IEEE Security & Privacy* 2016, <https://eprint.iacr.org/2015/1243.pdf>

Sugawara, T., Suzuki, D., Fujii, R., Tawa, S., Hori, R., Shiozaki, M., and Fujino, T.
Reversing Stealthy Dopant-Level Circuits, *International Conference on Cryptographic Hardware and Embedded Systems*, ser. CHES. New York, NY: Springer-Verlag, 2014, pp. 112–126.

Hicks, M., Sturton, C., King, S.T., and Smith, J.M.
Specs: A lightweight runtime mechanism for protecting software from security-critical processor bugs, *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS. Istanbul, Turkey: ACM, 2015, pp. 517–529.

Forte, D., Bao, C., and Srivastava, A.
Temperature Tracking: An Innovative Run-time Approach for Hardware Trojan Detection, *International Conference on Computer-Aided Design*, ser. ICCAD. IEEE, 2013, pp. 532–539.

Samuel Greengard is an author and journalist based in West Linn, OR.

© 2017 ACM 0001-0782/17/2 \$15.00

Milestones

ACM Recognizes 45 Distinguished Members

ACM recently named 45 new Distinguished Members in recognition of their contributions advancing the science, engineering, and education of computing, and which highlight the role of computing in shaping society today.

"The contributions of our Distinguished Members lead to breakthroughs that improve our lives, advance the frontiers of scientific discovery, and boost economic development," says ACM President Vicki L. Hanson. "Our global roster of 2016 Distinguished Members reminds us that excellence in our field knows no borders. For all our new Distinguished Members, we celebrate their dedication to computing, their creativity, and their exemplary professional accomplishments."

The new Distinguished Members have made contributions in areas including education, data privacy, security, networking, machine learning, distributed systems, multimedia computing, human-computer

interaction, programming languages, database management, information retrieval, computational biology, molecular computing, and software engineering.

2016 ACM DISTINGUISHED MEMBERS

DISTINGUISHED EDUCATORS:

Michael Clancy, *University of California, Berkeley*
Michelle Craig, *University of Toronto*
Amruth N. Kumar, *Ramapo College of New Jersey*
K.R. Venugopal, *University Visvesvaraya College of Engineering/Bangalore University*

DISTINGUISHED ENGINEERS:

David Carmel, *Yahoo Research*
Matthew L. Cooper, *FX Palo Alto Laboratory*
Rudra Dutta, *North Carolina State University*
Hubertus Franke, *IBM T.J. Watson Research Center*
Emden R. Gansner, *Google*
Tie-Yan Liu, *Microsoft Research Asia*

Heiko Ludwig, *IBM Research*
Jacquelyn Martino, *IBM*
David Ayman Shamma, *Centrum Wiskunde & Informatica*

DISTINGUISHED SCIENTISTS:

Joanne M. Atlee, *University of Waterloo*
Sonia Bergamaschi, *University of Modena and Reggio Emilia*
Raheem A. Beyah, *Georgia Institute of Technology*
Tevfik Bultan, *University of California, Santa Barbara*
Shigang Chen, *University of Florida*
Otfried Cheong, *Korea Advanced Institute of Science and Technology*
Shing-Chi Cheung, *Hong Kong University of Science & Technology*
Alberto del Bimbo, *University of Florence*
Josep Domingo-Ferrer, *Universitat Rovira i Virgili*
Sebastian Elbaum, *University of Nebraska-Lincoln*
Geraldine Fitzpatrick, *TU Wien*
Zhenjiang Hu, *National Institute of Informatics*
Gang Hua, *Microsoft Research*
Pan Hui, *Hong Kong University of Science and Technology*

Katherine Isbister, *University of California, Santa Cruz*
Murat Kantarcioglu, *University of Texas at Dallas*
Fabio Kon, *University of São Paulo*
Laks V.S. Lakshmanan, *University of British Columbia*
Stefano Lonardi, *University of California, Riverside*
Sanjay Madria, *Missouri University of Science and Technology*
Tao Mei, *Microsoft Research Asia*
Suman Nath, *Microsoft Research*
George Necula, *University of California, Berkeley*
Chong-Wah Ngo, *City University of Hong Kong*
Corina Pasareanu, *Carnegie Mellon Silicon Valley and NASA Ames Research Center*
Marian Petre, *The Open University*
Weisong Shi, *Wayne State University*
Prasun Sinha, *Ohio State University*
Darko Stefanovic, *University of New Mexico*
Yufei Tao, *Chinese University of Hong Kong*
Shuicheng Yan, *Qihoo/360 and National University of Singapore*
Yu Zheng, *Microsoft Research/Shanghai Jiao Tong University*

It's Not the Algorithm, It's the Data

In risk assessment and predictive policing, biased data can yield biased results.

CRIME IN THE U.S. has fallen dramatically over the past three decades, with 2014 statistics from the Federal Bureau of Investigation (FBI) noting the number of violent crimes committed per 100,000 people in 2013 (368) was less than half the level seen in 1991 (758).

Nevertheless, the debate continues over how to maintain these lower crime rates while addressing issues of fairness in the way communities are policed, as well as how to effectively and fairly use risk-assessment tools that can be relied upon by sentencing courts or parole boards.

There are two primary issues at stake: risk-assessment algorithms, which weigh a variety of factors related to recidivism, or the likelihood an individual will commit another crime and wind up back behind bars; and predictive policing, which has been described as using data analytics and algorithms to better pinpoint where and when a crime might occur, so police resources can be more efficiently deployed. Both issues are fraught with challenges—moral, logistical, and political—and opinions on whether they can be fairly and ethically utilized largely depend on how one views the nature of policing and the criminal justice system.

There is no debate that both of these types of technologies are being used on a fairly widespread basis in the U.S. According to a 2013 article published by Sonja B. Starr, a professor of law at the University of Michigan Law School, nearly every state has adopted some type of risk-based assessment tools to aid in sentencing. The primary concern related to these tools revolves around the use of computerized algorithms, which provide risk scores based on the result of questions that



Predictive policing systems identify “hotspots” where crime risk is the highest.

are either answered by defendants or pulled from criminal records, and whether such tools may ultimately penalize racial minorities by overpredicting the likelihood of recidivism in these groups.

The most widely known of these tools is COMPAS (Correctional Offender Management Profiling for Alternative Sanctions), a software tool owned by Northpointe, Inc., which has been used by a number of jurisdictions, including Broward County, FL, the State of New York, the State of Wisconsin, and the State of California, among others. The tool is seen as a success by many jurisdictions, such as New York State, which issued a 2012 report highlighting the effectiveness of the recidivism scale, noting, “the Recidivism Scale worked effec-

tively and achieved satisfactory predictive accuracy,” with an accuracy rate of 0.71 AUC (area under curve) value (the optimal AUC value is 1.0, which would indicate no false positives/all true positives were identified).

The report noted actual and expected rates for any re-arrest were closely aligned across scores, and that the tool was more effective with higher-risk cases (53.8% re-arrest rate for those deemed high-risk by the tool, versus 16.9% for those deemed low-risk by the tool).

Nevertheless, in recent years, there has been significant criticism from many in academia and a scathing investigative analysis from *ProPublica* (whose website describes it as “an independent, non-profit newsroom that

COMMUNICATIONS APPS

Access the latest issue, past issues, BLOG@CACM, News, and more.



Available for iPad, iPhone, and Android



Available for iOS, Android, and Windows

<http://cacm.acm.org/about-communications/mobile-apps>



Association for Computing Machinery

produces investigative journalism in the public interest”), which charged that the COMPAS algorithm and questions used to inform the algorithm were biased, since they relied on factors that could correlate with race. Critics say factors such as poverty, postal codes, and employment status can be used as proxies for race, as some are more highly correlated with minorities. Despite these limitations, the COMPAS tool survived its first major legal challenge in July 2016, when the Wisconsin Supreme Court ruled that judges can consider such risk scores during sentencing, but warnings must be attached to the scores to flag the tool’s “limitations and cautions.”

Moreover, the court specified that a computerized risk score cannot be the “determinative factor” in deciding whether someone is incarcerated or granted probation, and raised concerns about how many of its risk factors could be correlated with race.

For its part, Northpointe did not respond by press time to queries to address either the impact of the Wisconsin decision or criticism by academic or watchdog groups.

The use of algorithms in law enforcement is not limited to sentencing and parole cases. Many police departments around the country (including those in Seattle, WA, Richmond, VA, and Baltimore County, MD) are taking a more proactive approach to policing using analytics and algorithms, although these tools also are being targeted for incorporating what critics contend are data that has been tainted by years of racially motivated or biased policing strategies.

One such tool being used by a number of police departments is PredPol, developed by mathematicians at the University of California, Los Angeles, and Santa Clara University in close collaboration with crime analysts and patrol officers at the Los Angeles and Santa Cruz police departments. The tool uses three data points to provide predictions on where crime is likely to occur: past type of crime, place of crime, and time of crime. It does not use any personal information about individuals or groups of individuals in its crime predictions.

“We’re using algorithms that go through historical crime reports,” explains George Mohler, chief scientist at

Santa Cruz, CA-based PredPol, Inc. “We use that data to estimate risk. Whether it’s patrol cars or foot patrols or community policing, where they’re engaging the community in those areas, we’re providing those locations on a Google Map for the officers to allocate their resources.”

The company cites success in a number of jurisdictions, such as Alhambra, CA (a 32% drop in burglaries and a 20% drop in vehicle theft since deploying PredPol in January 2013), Los Angeles (the city’s Foothill division saw a 20% drop in predicted crimes year over year from January 2013 to January 2014), and Norcross, CA (a 15%–30% reduction in burglaries and robberies in the four months after it deployed the technology in August 2013).

“We’ve made the decision to not use [demographic or personally identifying information], partially because when you do use them, there’s a diminishing return on accuracy you get,” Mohler says. “Secondly, I think as a company, and with the agencies that use these tools, there is concern about these algorithms being biased.”

Another predictive policing tool being deployed by police departments is Motorola Solutions’ CommandCentral Predictive. This tool takes historical crime data (including exact locations, types of crimes, and times of day at which they were committed) and compares that data with a more recent snapshot of a particular area, which allows changes or anomalies to be easily identified.

Daniel (DJ) Seals, a former police officer and industry expert with Motorola Solutions, says CommandCentral incorporates a machine-learning algorithm that compares historical crime

PredPol uses three data points to provide predictions on where crime is likely to occur: past type of crime, place of crime, and time of crime.

data with more recent data to create a more accurate crime model and forecast, as opposed to simply relying on older data that may not be reflective of more recent activity.

Also, Seals says, CommandCentral introduces into the algorithm the concept of seasonality, which addresses crime patterns when temperatures rise or fall, further improving the granularity of the algorithm. Nonetheless, Seals agrees CommandCentral is a tool to help officers, not a replacement for the judgement of experienced officers.

“It takes a seasoned officer to look at the data, and say, ‘hey, I know what that is,’” Seals says. “It may be seemingly benign, but to that seasoned officer who knows the patterns, who knows the persons in that area, that sounds like ‘Bob.’ ‘Bob used to do that, and Bob just got out [of prison.]’”

Critics, however, say tools such as PredPol and CommandCentral are inherently biased since they rely heavily on reported crimes data, which is often concentrated in areas that are heavily policed, thereby skewing statistics to overrepresent poor or minority communities.

“We know that we have a history of racially biased policing in the United States, and that has fed into all the data that we have on where arrests have occurred, which crimes are more likely to occur in specific communities, and at which particular times,” says Jennifer Lynch, senior staff attorney at the Electronic Frontier Foundation. “That’s the data that’s being fed into predictive policing algorithms.”

Still, it is difficult to discount the value of event-based predictive policing, which relies on actual data on crimes that have been committed; ignoring this data could result in losing opportunities to prevent additional criminal acts.

“There has been a lot of research on near-repeat effects in crime,” PredPol’s Mohler says. “If someone breaks into a car in a certain neighborhood and is successful, they’ll often return to that same neighborhood a few days later, and break into another car.”

Systems such as PredPol and CommandCentral likely can spot such trends more quickly than relying on crunching historical crime statistics by hand, and allow law enforcement to target resources to address specific incidents.

Motorola’s Seals agrees, noting that

Critics say these tools are inherently biased since they rely on reported crimes data, which is often concentrated in heavily policed areas, skewing statistics to overrepresent the poor and minorities.

CommandCentral does not just rely on data from years ago. “As we get closer to the time we’re predicting, we actually crunch another shorter term [algorithm],” Seals says. What’s more, as it employs a learning algorithm, CommandCentral will get more accurate over time, if the system is properly updated.

Ultimately, however, “The algorithm itself may not be biased, but the data used by predictive policing algorithms is colored by years of biased police practices,” the EFF’s Lynch says, citing government statistics that up to 15% of vehicle thefts and 65% of rapes or sexual assaults are not reported, and noting that these non-reported crimes may be occurring in areas that are not necessarily deemed “high crime.”

“An algorithm can only predict crime based on the data it already has,” Lynch says. “This means it will continue to predict crime that looks like the crime we already know about, and will miss crimes for which we don’t have data.”

What’s more, defenders of predictive policing admit it must be accompanied by better community police outreach and transparency, to engender greater trust in these types of systems. Writing in *The Wall Street Journal* in April 2016, Jennifer Bachner, director of the master of science in government analytics program at Johns Hopkins University and author of a paper that supports greater use of predictive policing, cited a need for both greater technology utilization

and solid community policing strategies to reduce crime.

“Departments that adopt predictive-policing programs must at the same time re-emphasize their commitment to community policing,” Bachner wrote. “Officers won’t achieve substantial reductions in crime by holding up in patrol cars, generating real-time hot-spot maps. Effective policing still requires that officers build trust with the communities they serve.”

Most importantly, the tools put in place must be used. A RAND Corporation study focused on a predictive-policing pilot program deployed in 2013 and 2014 by the Chicago Police Department called Strategic Subjects List, which examined data on people with arrest records and generated a list of several hundred individuals deemed at elevated risk of being shot or committing a shooting.

While an analysis of the program found that people on the list were nearly three times as likely to be arrested for a shooting as those who did not get flagged by the system, the system resulted in very few arrests. This was due the presence of no fewer than 11 other violence-reduction programs in use at the time, so officers simply ignored the data, and their superiors did not make utilizing the system a priority. ■

Further Reading

Starr, S. B.

Evidence-Based Sentencing and the Scientific Rationalization of Discrimination (September 1, 2013). *Stanford Law Review*, Forthcoming; U of Michigan Law & Econ Research Paper No. 13-014. Available at SSRN: <http://ssrn.com/abstract=2318940>

New York State COMPAS-Probation Risk and Need Assessment Study: Examining the Recidivism Scale’s Effectiveness and Predictive Accuracy <https://www.ncjrs.gov/App/Publications/abstract.aspx?ID=269445>

Wisconsin v. Loomis July 2016 Decision: <https://www.wicourts.gov/sc/opinion/DisplayDocument.pdf?content=pdf&seqNo=171690>

Statistics on Non-Reported Crime:

Truman, J. and Langton, L. *Criminal Victimization*. September 29, 2015. U.S. Department of Justice. <http://www.bjs.gov/content/pub/pdf/cv14.pdf>.

Keith Kirkpatrick is principal of 4K Research & Consulting, LLC, based in Lynbrook, NY.

© 2017 ACM 0001-0782/17/2 \$15.00

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 60 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,



Vicki L. Hanson
President
Association for Computing Machinery



Association for
Computing Machinery

Advancing Computing as a Science & Profession

SHAPE THE FUTURE OF COMPUTING. JOIN ACM TODAY.

ACM is the world's largest computing society, offering benefits and resources that can advance your career and enrich your knowledge. We dare to be the best we can be, believing what we do is a force for good, and in joining together to shape the future of computing.

SELECT ONE MEMBERSHIP OPTION

ACM PROFESSIONAL MEMBERSHIP:

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

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.

Priority Code: CAPP

Payment Information

Name _____
ACM Member # _____
Mailing Address _____
City/State/Province _____
ZIP/Postal Code/Country _____
Email _____

Payment must accompany application. If paying by check or money order, make payable to ACM, Inc., in U.S. dollars or equivalent in foreign currency.

- AMEX VISA/MasterCard Check/money order

Total Amount Due _____
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

Return completed application to:
ACM General Post Office
P.O. Box 30777
New York, NY 10087-0777

Prices include surface delivery charge. Expedited Air Service, which is a partial air freight delivery service, is available outside North America. Contact ACM for more information.

Satisfaction Guaranteed!

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.



Association for
Computing Machinery

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
acm.org/join/CAPP

Inside Risks

The Future of the Internet of Things

The IoT can become ubiquitous worldwide—if the pursuit of systemic trustworthiness can overcome the potential risks.

AS SUGGESTED IN the previous *Communications Inside Risks* column (“Risks of Automation,” October 2016⁶), the Internet of Things (IoT) has the potential to encompass and instrument an enormous range of connected devices—including home appliances and utilities, wearables, homes and corporate buildings, industrial processes, medical devices, law-enforcement devices, military equipment, and other connected applications that today might be barely imaginable. In the present context, “Things” are simply those computerized and networked devices that become part of the IoT. Some of those Things will be directly accessible over the Internet, whereas others would be supposedly hidden in local networks behind firewalls and address-translating routers.

There are already many risks recognizably associated with the IoT. Some risks are old and well known, but exacerbated by the unprecedented scale of the IoT; estimates for the next few years suggest tens of billions of Things. Other risks may be new, stemming from

the nature of how these Things are designed, what they are used for, how they are deployed and managed (or not managed), and how market forces will influence the development. In this column, we outline some of those risks and what might need to happen if the IoT is to deliver the benefits envisioned for it—with a reasonable level of trustworthiness. Our message is intended as a wake-up call for computer professionals, but is also relevant to everyone involved as a user.

Security and privacy are both extremely important in the IoT, because the potential consequences of successful attacks could impact human lives and safety, and cause death and destruction—directly or indirectly. Privacy violations that let criminals exploit information about potential victims can also constitute threats to safety.

Things Turning Evil

A recent distributed denial-of-service (DDoS) attack⁷ has demonstrated the ubiquitousness of vulnerabilities in the current still-primitive Internet of Things. Many devices including

closed-circuit TV cameras, cable set-top boxes, and digital video recorders (DVRs) were compromised and used as unwitting botnet zombies. This significant event used malware (Mirai) that searches for vulnerable victims, and whose source code had been freely published. By targeting the DNS services provided by Dyn, this attack seriously interfered with user access to major services such as Twitter, Amazon, Tumblr, Reddit, Spotify, and Netflix. In one fell swoop, it exposed the tip of just one of many hazardous icebergs. While earlier DDoS attacks using Mirai had exploited hundreds of thousands of devices, this attack appeared to involve tens of millions of compromised devices—according to a statement from Dyn.¹³ The attack illustrates some of the risks associated with having very large numbers of inadequately protected Things connected to the Internet—particularly Things that are simple enough to be vulnerable to compromise, but sufficiently capable to be part of a distributed attack that floods the victims’ sites with seemingly legitimate requests. Note that the own-



ers or users of compromised devices are often not aware their devices are being used to attack other systems.

Vulnerabilities

Evidently, many of these devices that unwittingly contributed to that DDoS attack were not actually behind any sort of firewall, or else had weak default firewall configurations that were easily exploited. Furthermore, some of the Things infected by Mirai were themselves small-office or home-office routers. While Mirai specifically exploited hardcoded passwords for Telnet/SSH services that users could not disable, it is generally foolish to put all the blame on any one weak link, when almost everything is a potential weak link.

Today, almost every computer-related system is likely to be already compromised, or else easily misused. We have *weakness in depth and breadth*, not *strength in depth*. Therefore, many problems will need to be overcome to make the IoT viable. We consider some of those problems, and some possible remediations. Ultimately, we need a total-system perspective that address-

es the potential vulnerabilities in the devices, the alleged firewall security, the network connections, the cloud services (some not even known to the users), and the Internet itself, as well as all its users and would-be malfeasors. The IoT is not an entity per se—it encompasses all of these entities and inevitably depends on them.

We suggest this recent DDoS botnet episode is merely a harbinger of events to come. IoT risks in the future will be pervasive, including potential compromises of requirements relating to trustworthiness. Such requirements must address networkwide issues such as human safety, security, reliability, robustness, resilience, functional interoperability, seamless ease of installation and use, rapid automated remediation of serious flaws, personal as well as institutional privacy, human well-being, and much more.

Some Illustrative IoT Risks

Denial-of-service attacks are damaging, but the ability to subvert Things remotely for arbitrary manipulation must be considered particularly threat-

ening. Here are just a few examples of application areas where the use of IoT devices brings inherent risks:

- Hospitals and healthcare establishments tend to use devices that are already remotely controlled or accessible Things: patient monitors, body scanners, pacemakers, defibrillators, infusion pumps, main and auxiliary power, lighting, air conditioning, and much more.

- Critical infrastructure sectors such as electric power, oil, natural gas, manufacturing, and transportation use IoT devices as sensors and actuators for automation and remote monitoring and control. The controllers themselves may be Internet accessible.

- Self-driving and automation-assisted interconnected automobiles must clearly be considered as Things, especially in automated highways of the future. Recent demonstrations of the ability to remotely take over critical vehicle controls illustrate just a few of the risks.⁵

Unlike general-purpose computers, IoT devices may be more closely associated with the physical world. While there have so far been relatively

few cases where physical destruction has been intentionally caused through computer compromise, this is likely to be a risk of serious concern for the IoT. From the known cases of programs in the 1960s that could exercise disk arms to cause the drives to self-destruct, to the 2007–2010 Stuxnet attack that appeared to be designed to damage nuclear enrichment centrifuges (and reportedly succeeded), cyberphysical attacks have exploited vulnerabilities that are features rather than flaws. In addition to the Things that control switches, valves, and motors, many Things have batteries—which suggests the potential ability to remotely cause certain devices to overheat enough to cause a fire or explosion. If vehicles or medical devices are remotely taken over by malicious attackers, people could be injured or killed by someone clicking from anywhere on the Internet. Manipulation of sensors or insertion of misinformation could indirectly cause other health hazards by inducing chemical spills, disrupting energy systems, or misrouting vehicles. Thus, human safety must be a fundamental issue for many types of Things.

Another critical difference between IoT devices and general-purpose computers involves management. For a desktop computer, laptop, tablet, or smartphone, there are rich interactions between users and devices. Some notion of management also must exist: for corporate devices there are system administrators in important designated roles, while for personal devices the user is typically also the administrator. However, for IoT devices, there may be very little room for user interaction, and the concept of ‘management’ is unclear.

While operating systems and applications for general-purpose computers in desktop, laptop, tablet, or smartphone form factor tend to be easy to keep updated, many IoT devices are difficult or impossible for users to update. Some devices will remain in use for their entire lifetimes, precisely as delivered—unless they are recalled, discarded, or just forgotten. In some of those cases, security updates will be essentially impossible or extremely difficult. In other cases, devices may be directly accessible remotely over the Internet; any update mechanisms

must be secured so that attackers cannot subvert them and insert their own updates or attacks.

For Things that necessarily have interactions with human users, their small size typically will not allow for touchscreens or keyboards. Thus, they must either rely on another device such as a tablet or smartphone for interaction, or else use other emerging modes of interaction such as voice inputs. For voice interfaces, there are problems with linguistic ambiguities, and obvious privacy risks associated with ubiquitous devices that continuously record and process voice conversations, as well as interesting opportunities for replay or synthesized voice-command attacks from one device to another device. As already evident in advertising applications, audio interfaces could also be used for covert ultrasound communication, inaudible to humans.⁹

Whereas botnet attacks may typically be stopped by blocking the command and control servers that orchestrate the attacks, the individual IoT devices are still compromised, and could be pulled into a new botnet at any time. We are left with many questions. For example, who is responsible for fixing these devices? What incentive would the owner of a connected camera have for going through the trouble of updating its firmware if it seems to work just fine as it is? Who is liable when major disruptions occur? Is it the manufacturer, the vendor, the person or organization who deployed the device, the cloud or back-end communications provider, or the unwitting user of the device? Each of these alternatives entails its own set of risks.

Until recently, consideration of most of these risks has been dominat-

Another critical difference between IoT devices and general-purpose computers involves management.

ed by the competitive rush to market, with very few concerns for trustworthiness. This reality tends to cause security and privacy to be sadly neglected. Clearly, that must change, suggesting the advent of some serious far-sighted systemic considerations—especially where the risks might be greatest.

Confronting the Risks

We next attempt to outline some steps that might be desirable. As has been noted in past Inside Risks columns, we have a serious need for considering risks in the context of total systems. The Internet of Things requires a much deeper concern for total-system trustworthiness, in which the security of Things is only one aspect—especially because at the moment there is essentially no real security in computer systems and networks. This reality is clearly making the problems of assuring trustworthiness much more difficult.

We enumerate here just a few of the steps that might be helpful for developers, administrators, and users. However, we explicitly caution that this summary is only an essential beginning, and inherently incomplete. It may not be surprising that what is needed is more or less consistent with the series of National Academies’ Computer Science and Technology Board reports over the past several decades, including most recently.² In addition, NIST’s Special Publication 800-160, Computer Security Resource (Nov. 2016; <https://doi.org/10.6028/NIST.SP.800-160>), addresses important engineering aspects. Also, in the context of the IoT, we need to reemphasize many topics that have been discussed in the Inside Risks series more generally and that are highly relevant here.

Some IoT devices will have simple applications running on bare metal, that is, without general-purpose operating systems. Other Things might need simple operating systems focusing just on specialized requirements such as real-time guarantees, while yet others may require full-fledged operating systems. Thus, scalable hardware and software are likely to be useful for economic reasons and operational effectiveness. Implementations are likely to range from micro-operating systems on small processors to larger reprogrammable environments for centralized

control of Things for entire enterprises. Similarly, a range of development support is needed—from totally embedded as-delivered hardware with no possibility of software changes (except perhaps for recalls and possible remote updates) up to Things with flexible development environments and programming-language support. Thus, programming languages and compilers might need to encompass the very simple and the much more complex. Concerns for greater trustworthiness will be important, especially for embedding potentially unsecure applications into a nevertheless trustworthy environment.

Users generally lack expertise and patience, have limited ability to cope with complexity, and are unaware of corner cases. Consequently, the design and implementation of user interfaces for Things and their controllers will require special attention and care. These interfaces need to be seamlessly easy to use, intuitively self-evident, and friendly for those who are technologically impaired, as well as adequately configurable by everyone. Particularly problematical are easily managed Things that exist today (conventional light bulbs, toasters, and so on) whose computerization might render them completely unusable when they fail. Even worse might be mechanically fail-safe devices today that might no longer work manually. One such example might be a fully automated automobile whose doors cannot be opened from the inside if the battery dies or the car is under water, or perhaps a refrigerator door that cannot be opened because its Thing controller has crashed—or been hacked. Fail-sensible techniques will be essential.

The needs for seamless installation and integration are critical from the customers' viewpoint, but this should not be a motivation for ignoring security. One of the major risks here is the prevailing quest for simplicity—for example, just barely meeting the bar for compliance with standards and expectations, as well as poorly addressing needs for ease of installation and ease of use. Standards are needed to facilitate interoperable installations involving many different vendors' devices. Connection protocols should not be as simplistic and unsecure as they often are today.

Today's supposedly sage advice about how to deal with safety and security needs to be significantly upgraded.

Any local networks within a home or enterprise must be suitably isolated from the Internet and other outside connections—except where interactions are explicitly desired and adequate protection can be assured. Certain systems and Things will to some extent have to be resilient and resistant to insider misuse, although that may be less important to friendly homes than corporate entities. On the other hand, Internet firewalls must be much more impervious to outsider misuse than today. Ideally, fixed passwords and default encryption keys should be eschewed in devices—although they are far too common today, and indeed were exploited by the Mirai malware (as noted previously). Nevertheless, there will be cases where trustworthy updates cannot be achieved and recalls might be the only alternative. To enforce recalls, firewalls may need to recognize traffic from recalled and/or compromised Things, and block the communication to protect systems on the rest of the Internet.

Also, we must consider needs for oversight, consumer protection, regulation, and liability for flagrant violations that result in serious risks. As software makes its rapid transfer into our physical world through “smart” Things, we cannot afford to simply transfer the notion that software tends to be provided “as is”—without liability for the consequences of flaws. Electronic products that have the potential to hurt or kill people are typically subject to some form of government regulation and testing to protect consumers. When the safe operation of a product is dependent on its software

being secure and reliable, regulation will need to address those aspects of product safety. Also, the responsibilities of everyone involved need to be established and made clear. For example, if your home burns down because of a hacking attack on your IoT installation, or your negligence in failing to protect your technological devices, could your insurance companies deny coverage for known but unaddressed vulnerabilities, or even preexisting conditions?

In summary, we will need some meaningfully trustworthy hardware and software components, and much better development and deployment practices than we have at present—to enable the IoT to provide adequate human safety, security, reliability, usability, and satisfied users.

Some Specific Efforts

It is highly desirable to study a few types of Things as developing prototypes in research and development, and attempt to ensure that all reasonable risks have at least been addressed. We would benefit from a few very successful cases to pave the way for how this could be done in the future. The combination of system engineering, hardware and software engineering, and careful application development—perhaps with some formal analyses to provide better assurance—would be extremely valuable to everyone else competing in the IoT marketplace. Thus, a few well-designed, well-developed, and trustworthy systems that are well documented would provide wonderful examples for other developers. A step in that direction is the documented example of principled security design for a fictitious wearable fitness-tracking system that was produced by the IEEE's Center for Secure Design under the auspices of the IEEE Cybersecurity Initiative.¹²

It would also be very important to provide developers with the tools and knowledge to build security, privacy, reliability, and other aspects of trustworthiness into the systems that they build. This is particularly important for developers of IoT systems who may have even less security expertise than traditional software developers. We have recognized this need, and are involved in several efforts to address the situation—including the new IEEE Cy-

bersecurity Development Conference (IEEE SecDev),³ and a strategic independent R&D initiative at SRI International on IoT security and privacy.

Some Thoughts for the Future

Today's population displays a wide range when it comes to understanding computer technology, ability to use it, and to have access to it. We can't deny access to essential services to portions of the population by ignoring their inability to correctly use certain technologies. Above all, we have serious needs for better computer literacy in the entire population.

Many of the risks and needs discussed here are not just specific to the Internet of Things, and have commonalities with more general uses of computers. However, we must also consider self-driving vehicles as Things in the evolving automated highways, as well as automated airplanes—and treat them similarly in the same basic context. The very concept of the IoT brings us to a much more personal and visceral focus in its manifestations in homes, vehicles, and wearables, and in that sense it touches everyone to some extent. Even those who are unwilling may eventually be forced to buy IoT-enabled appliances, simply because there are no longer any alternatives.

Today's supposedly sage advice about how to deal with safety and security needs to be significantly upgraded. For example, while we are familiar with admonitions such the following, not everyone follows them: Beware of social engineering, hucksters, and easy solutions! Don't click on suspicious links! Don't display your most personal information on social media! Adhere to (or better yet, exceed) best practices for security! The new risks will be much more pervasive, and we will need to determine what reasonable caution and common sense will look like in the world of the IoT. Indeed, the IoT is likely to become very contentious unless serious coordinated efforts are made proactively by governments, standards committees, purveyors of Things and Thing infrastructures (including the Internet itself) and user communities. For considerable further background, please see recent testimony before the U.S. Congress.^{4,10} Also, some so-called best practices are considered in recommendations from the Department of Homeland Security¹¹ and BITAG.⁶

The future may be very murky unless proactive attention is paid to decide which Things can realistically be implemented wisely—and which might be simply too risky.

However, as we have noted in earlier Inside Risks columns, best practices are generally nowhere near good enough.

Considering the Keys Under Door-mats report,¹ the prospect of billions of sensor-equipped and Internet-connected IoT devices would be tempting to any organization that wants to collect information for intelligence or evidence, or to exploit the devices for propagating DDoS attacks, or other nefarious purposes. The risks of dumbing down cybersecurity and cryptography for such purposes would be enormous—especially with respect to the IoT.

There is much more on this topic than could be written here. However, this column is only an initial stake in the ground. Overall, there are no easy answers, but the time to begin asking the incisive questions is now.

Conclusion

We have described problems and potential risks that are associated with the evolving Internet of Things. It remains to be seen whether the IoT and its Things can *burgeon* (grow and flourish, as the way of the future), or *sturgeon* (sometimes surviving competitively for up to two decades if not caught), or be more like the female salmon (with very short lives once they spawn). In any case, we need much more than a *surgeon* to fix things (and Things). Incremental change is not likely to succeed (indeed, it has been ineffective for so many years), and some sort of radical change may be needed.

The future may be very murky unless proactive attention is paid to decide which Things can realistically be implemented wisely—and which might be simply too risky. We must then ensure that those beneficial Things can be integrated into the necessary total-system trustworthiness (which we do not yet have). Thus, we need to *urge on* to make the IoT truly usable, and then *surge on* to ensure that it happens with appropriate trustworthiness. **□**

References

1. Abelson, H. et al. Keys under doormats: Mandating insecurity by requiring government access to all data and communications. *Journal of Cybersecurity* 1, 1 (Nov. 17, 2015); Oxford University Press; <http://www.cybersecurity.oxfordjournals.org/content/1/1/69>
2. Computer Science and Technology Board, National Academies of Science, Engineering, and Medicine. *Foundational Science for Cybersecurity*, final report, 2017.
3. Cunningham, R. et al. IEEE SecDev 2016: Prioritizing Secure Development. IEEE Security and Privacy (July–Aug. 2016), 82–84. <https://www.computer.org/csdl/mags/sp/2016/04/msp2016040082.pdf>
4. Fu, K. Infrastructure Disruption: Internet of Things Security, Testimony before the U.S. House of Representatives Committee on Energy and Commerce, Subcommittee on Communications and Technology and Subcommittee on Commerce, Manufacturing, and Trade (Nov. 16, 2016); <https://energycommerce.house.gov/hearings-and-votes/hearings/understanding-role-connected-devices-recent-cyber-attacks>
5. Greenberg, A. Hackers remotely kill a Jeep on the highway—With me in it. *Wired* (July 21, 2015); <http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/>
6. Internet of Things (IoT) Security and Privacy Recommendations, BITAG Broadband Internet Technical Advisory Group, November 2016: [http://www.bitag.org/documents/BITAG_Report_-_Internet_of_Things_\(IoT\)_Security_and_Privacy_Recommendations.pdf](http://www.bitag.org/documents/BITAG_Report_-_Internet_of_Things_(IoT)_Security_and_Privacy_Recommendations.pdf)
7. Krebs on Security (Oct. 21, 2016); <https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/>
8. Neumann, P.G. Risks of automation: A cautionary total-system perspective of our cyberfuture. *Commun. ACM* 59, 10 (Oct. 2016); <http://www.csl.sri.com/neumann/insiderisks.html#240>
9. Newman, L.H. How to block the ultrasonic signals you didn't know were tracking you. *Wired* (Nov. 3, 2016); <https://www.wired.com/2016/11/block-ultrasonic-signals-didnt-know-tracking/>
10. Schneider, B. Testimony before the U.S. House of Representatives Committee on Energy and Commerce, Subcommittee on Communications and Technology and Subcommittee on Commerce, Manufacturing, and Trade (Nov. 16, 2016); <https://energycommerce.house.gov/hearings-and-votes/hearings/understanding-role-connected-devices-recent-cyber-attacks>
11. Strategic Principles for Securing the Internet of Things. Department of Homeland Security, along with an IoT Fact Sheet (Nov. 15, 2016); <https://www.dhs.gov/securingtheIoT>
12. West, J. et al. WearFit: Security Design Analysis of a Wearable Fitness Tracker, February 2016; <http://cybersecurity.ieee.org/blog/2016/02/17/wearfit-security-design-analysis-of-a-wearable-fitness-tracker/>
13. York, K. Dyn Statement on 10/21/2016 DDoS Attack (Oct. 22, 2016); <http://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/>

Ulf Lindqvist (ulf.lindqvist@sri.com) is a Program Director in the Computer Science Lab at SRI International.

Peter G. Neumann (peter.neumann@sri.com) is Senior Principal Scientist in the Computer Science Lab at SRI International, and moderator of the ACM Risks Forum.

Copyright held by authors.

Education

Fostering Creativity through Computing

How creative thinking tools and computing can be used to support creative human endeavors.

COMPUTING HAS THE potential to provide users opportunities to extend their *creative expression* to solve problems, create computational artifacts, and develop new knowledge. The pervasive nature of computing and accessibility of digital tools is also transforming K–12 education as students move from being mere consumers of content to engaging in the subject matter by creating computational artifacts. Take Scratch, for example, which is one of the many tools designed to teach kids to code, and comes with varying levels of support for educators implementing them in both formal and informal learning settings. Scratch provides students with an opportunity to express their creativity through stories, games, and animations. While Scratch has the potential to be a powerful tool, it is often used as little more than a presentation tool in the classroom. Studies of Scratch users show that few projects use variables or control flow data structures. While the Scratch environment provides a ‘low floor, high ceiling’ that allows beginners to dive into the environment without frustration, many students do not advance to a higher level. Tools like Scratch can empower students to showcase their creativity like never before; however, the way these tools are taught by teachers and used by students significantly influences whether students move along the creativity continuum.



A prototype implementation of Scratch Blocks, from Google Developers Blog, in collaboration with the MIT Media Lab's Scratch Team.

While Scratch is widely used, we know little about how it influences students' creative thinking.

K–12 Computer Science Education and Creativity

In his widely viewed TED talk, Sir Ken Robinson severely criticized educational institutions, claiming that we are “educating people out of their creativity.” Perhaps in response to this, or perhaps just due to the recognition of the importance of students developing creativity as they learn computer science, the College Board has identified creativity as one of the seven “big

ideas” as part of the new Advanced Placement CS Principles (CSP) course. In fact, variations of the word creative appear 62 times within the AP CSP Course and Exam Description. The argument is that computing fosters creativity by allowing individuals to move from merely being consumers of technology to building tools that can have a significant impact on society. The CSP course outlines how computing can enable people to not only use computing for creative expression, but also “extend traditional forms of human expression and experience.”¹ An understanding and use of computing

Computational Support for Academic Peer Review: A Perspective from AI

Learning from Mobile Technology

The Path to the Top: Insights from Career Histories of Top CIOs

Powering the Next Billion Devices with Wi-Fi

Reasoning on Data Partitioning for Single-Round Multi-Join Evaluation in Massively Parallel Systems

How to Make Computing Conferences, Digital Resources, and SIGs More Inclusive

Time, But Faster

Heterogeneous Computing: Here to Stay

Research for Practice: Distributed Transactions and Networks as Physical Sensors

Plus, the latest news on advances in speech synthesis, the future of semiconductors, and dark e-commerce.

(such as software tools and services), deep knowledge of a discipline, and creative expression allows individuals to create computational artifacts and/or solve problems. The partnership an individual establishes with computing tools enhances not only his/her creative expression, but it can even lead to new forms of artifacts.³ For example, musician Iannis Xenakis used probability distribution in the early 1950s to compose music, which he called Stochastic Music. In order to accelerate the stochastic calculations, Xenakis started programming. His programs not only computed the composition of the orchestra (percentages of each section), but also the assignment of a note to particular instrument. The deep knowledge of the discipline (music) and an understanding of computer programming allowed Xenakis to combine the power of computing to compose stochastic music.

The CS Principles framework highlights this creative aspect of what Xenakis was able to accomplish with computing—extending traditional forms of human expression and experience. However, it should be noted that in spite of the emphasis on creativity in the CS Principles framework, it is the only one of the seven big ideas not explicitly being tested as part of the new AP CSP exam. While we agree that creativity is a complex construct to understand and assess, we could learn from other domains such as psychology, where a number of creativity measures (for example, the Torrance Test of Creative Thinking, Guilford's Alternate Uses Test, Wallace and Kogan's test, and others) exist. Perhaps the College Board is developing a clear rubric for grading creativity in the CSP exam, as a means to measure creativity of students' submitted portfolios.

The question still remains how to expose students to these two prominent forces of creativity and computing within the context of particular disciplines, which can lead to solving ill-structured problems in the 21st century. How do we use computing across the various subject areas students encounter in their elementary and secondary spectrum of schooling? In the remainder of this column, we focus on how computing can be taught in a manner to enhance students' creative

Creativity involves a set of thinking tools that overlap with fundamentals of computer science, which can in turn support the development of creative thinking.

thinking. It is important to note that we are not new in proposing how to develop creativity in students. Nearly a century ago, Jacques Hadamard explored invention in mathematics (as an example of invention in general) to understand the processes great mathematicians use to invent.² The *Cambridge Handbook of Creativity* provides a comprehensive overview of creativity research from its relation to cognition to its domain specificity to assessing creativity. While there has been a great deal of work in trying to understand and grow the creative process, we are particularly intrigued by the work of creativity researchers like Robert and Michele Root-Bernstein, and in this column discuss how their work on creative thinking can inform computer science education.

Creative Thinking through Computer Science

Creativity involves a set of thinking tools that overlap with fundamentals of computer science, which can in turn support the development of creative thinking. Here, we outline some of the creative thinking tools, their overlap with computer science, and argue how computing can be used to support creative human endeavors.

Observing is one of the critical creative thinking tools that goes beyond sight to include hearing, smell, and taste, all of which allow us to acquire knowledge.⁴ Another related thinking tool for creativity is *imagining*, which allows individuals to visualize—to imagine the look of things that do not

physically exist. For example, physicist Richard Feynman used visual images as the solutions to a problem before ever jumping into mathematical equations for the answers. This ability to perceive by observing and imagining is critical to think creatively and innovate. Within computer science, perception plays a significant role for researchers working in visual computing areas, such as computer graphics and vision. For example, the Graphics Vision Visualization group at Trinity College (<http://gv2.cs.tcd.ie>) draws extensively on how humans perceive when developing visualizations, such as virtual agents. Access to such tools allows students to think in powerful ways and help awaken their creative thinking skills related to perceiving. We could enhance students' perceiving skills and extend this notion of imaging by having them imagine how algorithms and code are structured and how they execute, that is leap back and forth between describing an algorithm and writing the code itself.

The ability to *abstract*—reducing information and detail in order to focus on concepts relevant to solve problems—is another essential creative thinking tool and its importance in computer science is highlighted by the fact that it is also one of the big ideas for the CS principles course. For computer scientists, abstraction is a fundamental

concept. In our Alice workshops with teachers, we motivate the need for decomposition and the use of methods (as a form of abstraction) by developing a solution to a problem without the use of abstraction. In our case, we teach a dragon to fly through the use of the primitive move, turn, and roll methods on each of the appropriate parts of the dragon's body. The code that has not been decomposed into smaller parts is very difficult to reason about, or to later modify. Decomposition through the use of methods allows the programmer to think about the method at a higher level of abstraction. Once the programmer has gotten the details of a method to work, it is no longer (generally) necessary to think about the detailed instructions that make up the method. The method simply works as it is supposed to work. We note that our approach is quite similar to the method taken by Mehran Sahami in his iTunes University Programming Methodology course, with a Karel the Robot task. Parameters can be taught in a similar manner. In our Alice example, rather than creating one method for the dragon to fly to the knight, and a second method for the dragon to fly to the king, it is possible to parameterize the target of the dragon's flight.

Finally, patterning is another thinking tool central to creativity and includes both the ability to recognize

patterns as well as being able to form patterns.⁴ Recognizing patterns plays a significant role for computer scientists working in the area of machine learning, especially when it comes to the extracting the right information based upon identifying specific patterns in large datasets. Similarly, forming patterns is key to putting all the pieces of information together in a scientific visualization. Students could, for example, use computing tools to learn about identifying and forming patterns in data first beginning with spreadsheets in elementary grades to using mathematical/statistical functions in Python in secondary school.

Conclusion

In this column, we have argued that computing provides students with a powerful mechanism to support their creative thinking. However, we need to carefully consider how we use the affordances of computing tools rather than merely putting them in front of teachers and their students. We need to address how teachers and their students use digital tools to engage in creative thinking skills as discussed in this column. We also need to develop measures that allow us to evaluate whether and how computing supports processes that aid students' creativity. We end with a hope. We hope that well-planned use of these computing tools will lead to more creative results than the results obtained by giving an elephant a paintbrush and a palette, as shown in the accompanying photo. While elephant art is unique, it does not illustrate, at least in our view, any degree of creativity by the elephant. ■

References

1. AP computer science principles draft curriculum framework: 2014; <https://advancesinap.collegeboard.org/stem/computer-science-principles>.
2. Hadamard, J. *The Psychology of Invention in the Mathematical Field*. Dover Publications, 1954.
3. Mishra, P. and Yadav, A. Of art and algorithm: Rethinking technology and creativity in the 21st century. *TechTrends* 57, 3 (2013), 10–14.
4. Root-Bernstein, R.S. and Root-Bernstein, M.M. *Sparks of Genius*. Houghton Mifflin, Boston, 1999.

Aman Yadav (ayadav@msu.edu) is an Associate Professor in the College of Education and Director of the Masters of Arts in Educational Technology program at Michigan State University, East Lansing, MI.

Steve Cooper (scooper22@unl.edu) is Associate Professor in the Department of Computer Science and Engineering at the University of Nebraska-Lincoln, and Director of the Jeffrey S. Raikes School of Computer Science and Management.

Copyright held by authors.



Elephant art: Unique, but not creative.



Kode Vicious

The Unholy Trinity of Software Development

Tests, documentation, and code.

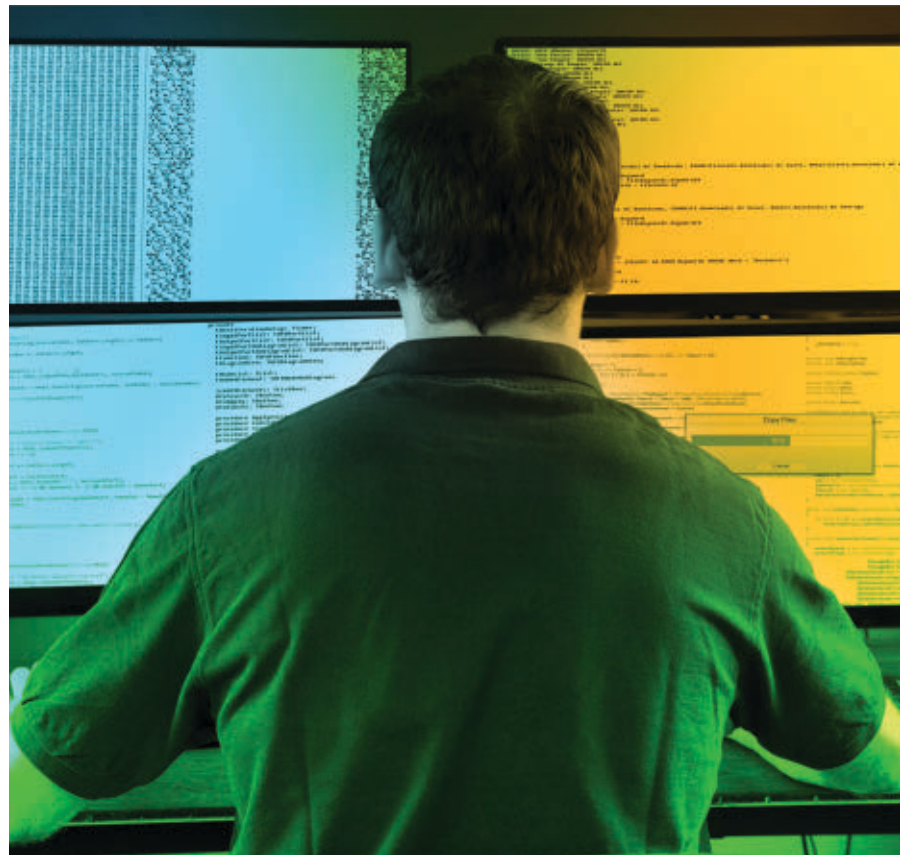
Dear KV,

I have read your columns for a few years and you have written a few times about testing and documentation, but you have not written about the relationship between the two. Many large projects, including the one I am working with now, keep their tests and documentation separate from the code and from each other. I have argued with my team about putting both of these closer to, or even embedded in, the source code, but there are always a few engineers who refuse to allow anything but code and comments in the source files. Their argument is about maximizing the screen space for code context, but that seems like a more reasonable argument for a 25-line terminal than for a modern development environment. What arguments would you make to get this software team to adopt a more holistic approach to the system we are working on?

System Integrationist

Dear SI,

Questions like this bring to mind the fact that source code, documentation, and testing are the unholy trinity of software development, although many organizations like to see them as separate entities. It is interesting that while many groups pay lip service to “test-driven development,” they do not include documentation in TDD.



I think you have found one of the reasons many developers fight against the integration of tests and documentation into source code, either directly into the source files or even close by in the same directory. The arguments against such integration include the one you mentioned about vertical space for code. Let's

treat that one first.

Software developers like new toys. Of course they do: they work on computers and computers are toys to us, and everyone likes things that are shiny. If you visit a modern software company, what do you see besides a sea of Aeron chairs? Lots and lots of monitors, and many of those are of

the 4K variety, meaning that a text editor, even with a large font, will give you more than 100 lines of code to look at—a 400% increase over the 80x25 monitors used to write code since the 1970s.

There is a school of thought that says a function or method of 100 lines is too long and should be broken down into smaller, more easily digestible chunks. If you work in such an environment, then there is no reason for the developers to argue against documentation or tests being integrated within the code itself, as a 25-line function can have 10 lines of documentation above it, and a reasonable documentation system, such as Doxygen, can assemble this set of “documentettes” into a greater whole. That greater whole, by the way, needs to be reviewed by someone who can turn it into language suitable for others to read. The worst code documentation is the kind that is not kept in sync with the code itself. The second worst type of such documentation is where the engineer writes something obvious, `foo()` function returns bar, which is easily seen from reading the code.

The best type of this documentation explains what the function does, what its possible error conditions are, and what conditions are required when the code is called. Multithreaded applications really need to have the locking requirements described in such documentation blocks. For some products such as end-user-facing systems, these document blocks will not generally find their way into the final manual, but they will be very useful to the person who is responsible for writing that manual. Libraries and other systems that are consumed by other programmers absolutely must have this style of documentation for the preservation of the sanity of all involved.

On the integration of tests into the source code, well, KV may be a bit old fashioned, but I do see that this is a tad harder and probably requires the source code to have special features embedded in it, or that the source-code editing system have special features, or both. Even with more than 100 lines of vertical space in which to code and document, adding any significant number of conformance tests will definitely

While many groups pay lip service to “test-driven development,” they do not include documentation in TDD.

dwarf the code and make each source file quite large and unwieldy.

Code folding, a common feature of many text editors, may help if you really are hell-bent on keeping the unholy trinity together in one file. The top of each source file would include the overarching documentation, a large block that describes the module and its purpose within the system. The source code would then be placed between the class/method/function documentation, and the conformance tests for the code would come last.

The main complaint you will encounter with the folding method is that it requires special characters and a smart code editor, although even Vim has code folding at this point. Since folding in the source file usually requires special tags, it would make sense to standardize these for the whole project so that there is one tag each for documentation, tests, and source code.

One advantage of combining all of these things is that the tools you use—including compilers, editors, test frameworks, and documentation extractors—can all point at the same directory hierarchy. Keeping tests, documentation, and code separate complicates the search paths for the various tools and leads to errors, such as pointing at the wrong place and getting the wrong tests, or similar problems.

Bringing together these three components so that they are easily developed in concert has a lot of advantages, but you are still going to have to help people past the mind-set of the terminal. These arguments often bring to mind the following scene from Neal

Stephenson’s *Snow Crash*, in which he describes how the main character, annoyingly named Hiro Protagonist, actually writes code: “... where everything that you see in the Metaverse, no matter how lifelike and beautiful and three-dimensional, reduces to a simple text file: a series of letters on an electronic page. It is a throwback to the days when people programmed computers through primitive teletypes and IBM punch cards.”

The next time your teammates complain about vertical space wasted on tests or documentation, hand them a punch card.

KV

Dear KV,

In the past 10 years I have noticed that the number of CPU cores available to my software has been increasing, but that the frequency of those cores is not much more than it was when I left school. Multicore software was a big topic when the trend first began, but it does not seem to be discussed as much now that systems often have six or more cores. Most programmers seem to ignore the number of cores and write their code as they did when systems had only a single CPU. Is that just my impression, or does this mean that maybe I picked the wrong startup this year?

Core Curious

Dear Core,

The chief contribution of multicore hardware to software design has been to turn every system into a truly concurrent system. A recently released digital watch has two cores in it, and people still think “digital watches are a pretty neat idea” (as in Douglas Adams’s *The Hitchhiker’s Guide to the Galaxy*). When the current crop of computer languages was written, the only truly concurrent systems were rare and expensive beasts that were used in government research labs and other similarly rarefied venues. Now, any clown can buy a concurrent system off the Internet, install it in a datacenter, and push some code to it. In fact, such clowns can get such systems in the cloud at the push of a button. Would that software for such

INTERACTIONS



ACM's *Interactions* magazine explores critical relationships between people and technology, showcasing emerging innovations and industry leaders from around the world across important applications of design thinking and the broadening field of interaction design.

Our readers represent a growing community of practice that is of increasing and vital global importance.



To learn more about us, visit our award-winning website <http://interactions.acm.org>

Follow us on Facebook and Twitter  

To subscribe: <http://www.acm.org/subscribe>



The purpose of software is to process data and therefore to take things and change them or to mutate state.

systems were as easily gotten!

Leaving aside the fact that most applications are now distributed systems implemented on many-core communicating servers, what can we say about the concurrent nature of modern software and hardware? The short answer is, “It’s all crap,” but that is not helpful or constructive, and KV is all about being helpful and constructive.

From our formative computer science years, we all know that in a concurrent system two different pieces of code can be executing simultaneously, and on a modern server, that number can easily be 32 or 64 rather than just two. As concurrency increases, so does complexity. Software is written to be executed as a set of linear steps, but depending on how the software is written, it may be broken down into many small parts that might all be running at the same time. As long as the software does not share any state between the concurrent code, everything is fine—well, as fine as any other nonconcurrent software. The purpose of software is to process data and therefore to take things and change them or to mutate state. The number of significant software systems that do not wind up sharing state between concurrent parts is very, very small.


Software that is written specifically without concurrency is, of course, easier to manage and debug, but it also wastes most of the processing power of the system on which it runs, and so more and more software is being converted from nonconcurrent into concurrent or being written for concur-

rency from scratch. For any significant system, it is probably easier to rewrite the software in a newer, concurrency-aware language than to try to retrofit older software with traditional concurrency primitives.

Now, I am sure you have read code that looks to be nonconcurrent—that is, it does not use threads in its process—and you might think that was fine, but, alas, nothing is ever really fine. Taking a collection of programs and having them share data through, for example, the file system or shared memory, a common early way of having some level of concurrency, does not protect the system from the evils of deadlock or other concurrency bugs. It is just as possible to deadlock software by passing a set of messages between two concurrent processes as it is to do the same sort of thing with Posix threads and mutexes. The problems all come down to the same things: idempotent updates of data and data structures, the avoidance of deadlocks, and the avoidance of starvation.

These topics are covered in books about operating systems, mostly because it was operating systems that first had these challenges. If, after this description, you are still curious, I recommend picking up one such book so that you at least understand the risks of concurrent systems and the land mines you are likely to step on as you build and debug, and debug, and debug such systems.

KV

 Related articles on queue.acm.org

Hickory Dickory Doc

Kode Vicious

<http://queue.acm.org/detail.cfm?id=2791303>

Microsoft's Protocol Documentation Program: Interoperability Testing at Scale

Nico Kicillof, Wolfgang Grieskamp, and Bob Binder

<http://queue.acm.org/detail.cfm?id=1996412>

Verification of Safety-critical Software

B. Scott Andersen and George Romanski

<http://queue.acm.org/detail.cfm?id=2024356>

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.

Privacy and Security

User-Centric Distributed Solutions for Privacy-Preserving Analytics

How can cryptography empower users with sensitive data to access large-scale computing platforms in a privacy-preserving manner?

FOR OVER A year, a high-profile initiative spearheaded by the City of Boston and the Boston Women's Workforce Council (BWWC) strived to identify salary inequities across various employee gender and ethnic demographics at different levels of employment, from executive to entry-level positions.¹¹ While the effort was supported by a diverse set of more than 100 employer organizations in the city—including major corporations, small businesses, and public/non-profit organizations—it was stalled by concerns about the confidentiality of the data to be collected in order to calculate aggregate metrics.²

A key enabling technology that allowed this effort to move forward was a Web-based application (which can be seen at 100talent.org) that we designed and implemented at Boston University to support the aggregation of sensitive salary data using secure multi-party computation (MPC).⁸ This application was used in a first-of-its-kind collaborative effort to compute aggregate payroll analytics without revealing the individual datasets of contributing organizations. This deployment of MPC, which received significant media attention,^{2,15} finally enabled the BWWC to conduct their analysis and produce a report representing their findings.⁴

MPC privately shards users' sensitive data across multiple servers in such a way that analytics may be jointly computed and released while ensuring that (small collections of) servers cannot learn any user's data. Theoretical constructs for MPC have been known for 35 years, with several existing software frameworks designed over the past 10 years.^{7,9}

MPC techniques can possess substantial social value: they enable society to benefit from collective data aggregation and analysis in contexts where the raw data is encumbered by legal and corporate policy restrictions on data sharing. Other examples of deploying MPC for social good include tax fraud detection³ and disease surveillance.⁵ Additionally, because MPC decouples computing and networking resources from data, users can leverage the benefits of large data centers without ceding control over their sensitive data.

However, MPC's social benefits cannot be realized unless we empower participating organizations (that is, their executives, directors, and legal advisors) with a clear, confident understanding of exactly how MPC protects their sensitive data and mathematically guarantees compliance with data sharing restrictions. The design and implementation of our own unique MPC platform was informed by nearly two years' worth of discussions with

non-technical personnel (including CIOs, CTOs, HR executives, and lawyers from key participating organizations), social scientists, and members of the city council that commissioned the study.¹³ These discussions had to take place in meetings and teleconferences where the only aids were whiteboards and slideshows; they involved both describing secret sharing in a concrete, hands-on way as well as providing details of the implementation and how it realized the capabilities and guarantees of this technique. Ultimately, these exchanges were necessary to demystify MPC for decision makers and, more generally, to help us understand and mitigate what we have come to realize are the hurdles that face real-world MPC deployments.

The systems community has grappled recently with the realization that its significant body of work on scalable platforms did not adequately consider the question of what minimum distributed computing configuration outperforms a single thread (COST).¹⁰ Analogously, in this column we argue that the extensive body of MPC research to date has not adequately considered the needs and circumstances of the ultimate users of MPC. Our own experience echoes and confirms thoughts expressed by other researchers in the community:¹⁶ "Secure computation is a general scheme; in reality one has to

choose an application, starting from a very real business need, and build the solution from the problem itself choosing the right tools, tuning protocol ideas into a reasonable solution, balancing security and privacy needs vs. other constraints: legal, system setting, etc.” We draw from our experience to advocate for the design of platforms that address concerns along Usability, Scalability, Entrustment, and Risk (USER) dimensions.

Usability

To meet the needs of our users, we rejected the most algorithmically expressive MPC solutions available in the literature.⁷ Instead, we found that what we needed was the simplest of protocols: just expressive enough for the application at hand while being comprehensible enough to fuel adoption among corporate officers, legal representatives, and rank-and-file employees. We also found that participants’ software platform and IT infrastructure inflexibilities and limitations (legacy systems, restrictive policies, firewalls, and so on) required the most lightweight solution: a simple browser-based application that could accommodate the familiar look and feel of a spreadsheet, with transparent open source code to enable outside auditing. Finally, our MPC protocol needed to accept contributors’ data asynchronously to simplify coordination and idempotently to allow contributors to fix errors.

Usable MPC is an enabling technology with substantial potential for social good, but only if enough participants are willing to contribute toward the analysis. In the pay equity scenario, the usability of both the protocol and its implementation helped decision makers—after only a few conversations—gain confidence in their understanding of the technology, appreciate that it would impose no significant burdens on their staff and infrastructure, and assured that features such as idempotence and asynchrony would make deployment logistically feasible and likely to produce meaningful results. This, in turn, increased the willingness of participants to contribute their sensitive data.

Usability also extends to the specification of policies governing proper uses of data. Existing MPC frameworks neglect to address privacy policies,

in part because the policy may not be expressible by either the original data contributor (who may lack expertise in privacy-related matters) or the analyst (who doesn’t know the users’ preferences or other uses of the data). Existing techniques from the programming languages research and formal methods communities such as policy-agnostic programming (in which the policies that govern inputs are specified independently from the dataflows and logic of the algorithm), as well as static analysis (to automatically derive policies from algorithms and compare them to user-specified policies) can play a significant role in validating whether an analytic is compatible with a specified privacy policy.

Scalability

Typically, MPC frameworks are evaluated based on their computational efficiency for simple analytics over relatively small datasets. This is a situation in which all modern frameworks perform rather well (that is, seconds to minutes).¹

However, human time dominates computing time in scenarios involving small-scale data such as the pay equity effort, in which a window spanning multiple days may be required to collect salary data from a large number of contributors operating according to incompatible schedules, rendering the computing time negligible by comparison. In this case, MPC frameworks should prioritize software development and IT infrastructure design over the speed of computing the analytic. At the other extreme, when aggregating large-scale datasets, an MPC framework should optimize the computation that can be performed locally so as to minimize the costs incurred due to MPC.

To resolve both challenges, we have integrated existing MPC frameworks into the Musketeer big data workflow manager.⁶ Whereas prior MPC frameworks require that software engineers design analytics in a domain-specific language, we permit rapid development in the well-known SQL and MapReduce paradigms, with automated generation of code to execute in existing back-end distributed frameworks like Hadoop, Spark, or Naiad so that developers and administrators can “focus on the what rather than the how of security.”¹² Ad-

ditionally, our framework automatically infers when sensitive data crosses trust boundaries in order to minimize usage of MPC. We tested this system to compute a market concentration metric over 160GB of public NYC taxi trips’ fare information with just 8.3% overhead over the corresponding insecure computation.¹⁴

Entrustment

At its heart, MPC permits a federation of trust among several computing entities such that each user only needs to trust that any one of them (or a small fraction) is honest. Most existing MPC research papers and software frameworks envision homogeneous entities. By contrast, we design a more flexible MPC framework that allows contributors to entrust entities with different responsibilities.

Along these lines, we provide a taxonomy of roles for entities that participate in MPC: a large, potentially a priori unknown number of *contributors* with private data; an *analyzer* who specifies an analytic; a publicly accessible *service provider* who collects encoded data from the contributors without requiring them to be online simultaneously and who also participates in the distributed computation; additional servers who participate in the distributed computation; one or more repositories that host the secure computing software; and the recipients of the analysis. Behind the scenes, there may also be privacy experts and software engineers who assemble one or more of the components in this ecosystem. In practice, parties using MPC may take on several of these roles simultaneously.^a MPC provides the recipients with the results of the analytic over the contributors’ data, and it provably guarantees that nobody learns anything else.

Just as each entity has different assignments, so too might they have different levels of trust in one another. For brevity, we focus here on the service provider, who must connect to all oth-

a Some readers may be familiar with a related technology: fully homomorphic encryption (FHE). Abstractly, FHE can be viewed as a specialization of MPC to the two-party outsourcing setting in which the contributor, analyzer, and recipient are the same party and in which the service provider’s computation does not require interaction.¹⁷

The empowering and enabling aspects of MPC will make substantial contributions to data-driven analysis and policymaking.

er entities and may require immense computing power. When both of these characteristics simultaneously apply, the service provider has a large attack surface and is well suited to being run within a cloud computing datacenter.

Our pay equity software enables the most powerful computing entity also to be the *least trusted*. Our service provider runs on Amazon Web Services to collect and store encoded data; however, contributors can choose instead to entrust the BWWC to protect the confidentiality of their data. We envision a future in which cloud providers offer ‘secure computing-as-a-service’ deployments of MPC that decouple control over data from computing power.

Risk

MPC research studies four types of adversaries: semi-honest entities who execute software as provided but may attempt to glean information along the way, covert adversaries who cheat only if they are unlikely to be caught, rational adversaries who cheat as long as the expected payout is larger than the expected penalty if caught, and fully malicious entities who perform any action necessary to breach the confidentiality or integrity of honest users.

We advocate for the MPC community to match cryptographic models of adversarial behavior with the economic (for example, reputation-based) and legal incentives that real-world users face. A more accurate and fine-grained characterization of risks can result in a faster, simpler MPC protocol that satisfies users’ needs. Our pay equity project exposed delicate economic and legal concerns whose impact upon risk


models should be explored further.

First, the existing risk models fail to capture the subtlety of reputation-based economic incentives. In the pay equity scenario, the analyzer and repository have the capacity to alter the software to leak secrets; however, they should not execute this capability due to the long-term damage to their reputation and economic viability. Analogously to the differences between the oneshot and iterated prisoner’s dilemma games, the rational model of MPC provides an incomplete view because it focuses on a single execution.

Second, MPC has a complex interconnection with the law. In our pay equity scenario, even if the BWWC could somehow learn the contributors’ data by cheating, it has a strong legal incentive not to acquire this data because it could then be exposed to lawsuits. Indeed, one of the major hurdles that faced BWWC prior to their use of our solution was the unwillingness of any single entity (including a major local university, originally enlisted to perform the study) to assume the liability in case of leakage or loss of data entrusted to them. Moreover, following MPC honestly may provide BWWC legal protections afforded by following best practices or by restricting data sharing. Hence, the BWWC has a strong legal incentive to act in a semi-honest manner. Conversely, appropriately written legal contracts can enshrine MPC’s constraints (for example, operating in the best interest of another entity, or forbidding collusion between entities) with enforceable civil penalties. We propose a greater examination of the implications of the law upon MPC and vice versa.

Conclusion

We are convinced that the empowering and enabling aspects of MPC will make substantial contributions to data-driven analysis and policymaking by enabling individuals and organizations at all levels to derive insights about their collective data without requiring that they share that data, but only if the technology is accessible both conceptually and technologically to a broad audience. In this column, we proposed a four-pronged research agenda to make MPC more usable along a variety of dimensions, increase its scalability for humans and computers alike, assign respon-

sibilities that align with existing trust relationships, and systematically understand the legal and economic risks when trust is violated. These recommendations are informed by our prior work deploying MPC to aggregate wage data and compute pay equity metrics—work that is, in the words of BWWC co-chair Evelyn Murphy, “beginning to show how to use sophisticated computer science research for public programs.”¹⁵ 

References

1. Archer, D.W. Maturity and performance of programmable secure computation. *IACR Cryptology ePrint Archive*, (1039), 2015.
2. Barlow, R. Computational thinking breaks a Logjam: Hariri Institute helps address Boston’s male-female pay gap. (Apr. 27, 2015); *BU Today*.
3. Bogdanov, D. *How the Estonian Tax and Customs Board Evaluated a Tax Fraud Detection System Based on Secure Multi-party Computation*. Springer, Berlin, Heidelberg, 2015, 227–234.
4. Boston Women’s Workforce Council Report 2016; <http://bit.ly/2iR2KhW>
5. El Emam, K. A secure protocol for protecting the identity of providers when disclosing data for disease surveillance. *Journal of the American Medical Informatics Association* 18, 3 (May 2011), 212–217.
6. Gog, I. Musketeer: All for one, one for all in data processing systems. In *Proceedings of the Tenth European Conference on Computer Systems (EuroSys)*, (2015), 2:1–2:16.
7. Hamlin, A. Cryptography for big data security. In Fei Hu, Ed., *Big Data: Storage, Sharing, and Security*. CRC Press, May 2016.
8. Lapets, A. *Secure Multi-Party Computation for Analytics Deployed as a Lightweight Web Application*. Technical Report BUCS-TR-2016-008, CS Dept., Boston University, July 2016.
9. Lindell, Y. and Pinkas, B. Secure multiparty computation for privacy-preserving data mining. *The Journal of Privacy and Confidentiality* 1 (2009), 59–98.
10. McSherry, F. Scalability! But at what COST? In *Proceedings of the 15th Workshop on Hot Topics in Operating Systems (HotOS XV)*, Kartause Ittingen, Switzerland (May 2015). USENIX Association.
11. 100% Talent: The Boston Women’s Compact; <http://bit.ly/YWryu2>
12. Shen, E. Cryptographically secure computation. *IEEE Computer* 48, 4 (2015), 78–81.
13. Signers of 100% Talent: The Boston Women’s Compact; <http://bit.ly/2bN48QJ>
14. Volgushev, N. DEMO: Integrating MPC in big data workflows. In *Proceedings of CCS 2016: The 23rd ACM SIGSAC Conference on Computer and Communications Security*, 2016.
15. Will Data Help Close The Gender Pay Gap?; <http://wbur.fm/2hdbjXa>
16. Yung, M. From mental poker to core business: Why and how to deploy secure computation protocols? In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS ’15*, ACM, 2015.

Azer Bestavros (best@bu.edu) is Professor of Computer Science and Founding Director of the Hariri Institute for Computing at Boston University.

Andrei Lapets (lapets@bu.edu) is a Research Scientist and Director of Research Development at the Hariri Institute for Computing at Boston University.

Mayank Varia (varia@bu.edu) is Research Scientist and Co-Director of the Center for Reliable Information Systems and Cyber Security at the Hariri Institute for Computing at Boston University.

This material is based upon work supported by the National Science Foundation under Grants No. 1414119 and 1430145.

Copyright held by authors.

Viewpoint

Smart Machines Are Not a Threat to Humanity

Worrying about machines that are too smart distracts us from the real and present threat from machines that are too dumb.

CONCERNS HAVE RECENTLY been widely expressed that artificial intelligence presents a threat to humanity. For instance, Stephen Hawking is quoted in Cellan-Jones¹ as saying: “The development of full artificial intelligence could spell the end of the human race.” Similar concerns have also been expressed by Elon Musk, Steve Wozniak, and others.

Such concerns have a long history. John von Neumann is quoted by Stanislaw Ulam⁸ as the first to use the term *the singularity*^a—the point at which artificial intelligence exceeds human intelligence. Ray Kurzweil⁵ has predicted that the singularity will occur around 2045—a prediction based on Moore’s Law as the time when machine speed and memory capacity will rival human capacity. I.J. Good has predicted that such super-intelligent machines will then build even more intelligent machines in an accelerating ‘intelligence explosion.’⁴ The fear is that these super-intelligent machines will pose an existential threat to humanity, for example, keep humans as pets or kill us all¹⁰—or maybe humanity will just be a victim of evolution. (For additional information, see Dubhashi and Lappin’s argument on page 39.)

I think the concept of the singularity is ill conceived. It is based on an oversimplified and false understanding of



intelligence. Moore’s Law will not inevitably lead to such a singularity. Progress in AI depends not just on speed and memory size, but also developing new algorithms and the new concepts that underpin them. More crucially, the singularity is predicated on a linear model of intelligence, rather like IQ, on which each animal species has its place, and along which AI is gradually advancing. Intelligence is not like this. As Aaron Sloman, for instance, has successfully argued, intelligence must be modeled using a multidimensional space, with many different kinds of intelligence and with AI progressing in many different directions.⁶

AI systems occupy points in this

multidimensional space that are unlike any animal species. In particular, their expertise tends to be very high in very narrow areas, but nonexistent elsewhere. Consider, for instance, some of the most successful AI systems of the last few decades.

► **Deep Blue** was a chess-playing computer, developed by IBM, that defeated the then-world champion, Garry Kasparov, in 1996. Deep Blue could play chess better than any human, but could not do anything other than play chess—it could not even move the pieces on a physical board.

► **Tartan Racing** was a self-driving car, built by Carnegie Mellon University and General Motors, which won

a https://en.wikipedia.org/wiki/Technological_singularity

the DARPA Urban Challenge in 2007. It was the first to show that self-driving cars could operate safely alongside humans, and so stimulated the current commercial interest in this technology. Tartan Racing could not play chess or do anything other than drive a car.

► **Watson** also developed by IBM, was a question answering system that in 2011 beat the World champions at the “Jeopardy!” general-knowledge quiz game. It cannot play chess or drive a car. IBM is developing versions of Watson for a wide range of other domains, including healthcare, the pharmaceutical industry, publishing, biotechnology, and a chatterbox for toys. Each of these applications will also be narrowly focused.

► **AlphaGo** was a Go-playing program, developed by Google’s DeepMind, that beat the World-class player, Lee Sedol, 4–1 in October 2015. AlphaGo was trained to play Go using deep learning. Like Deep Blue, it required a human to move the pieces on the physical board and could not do anything other than play Go, although DeepMind used similar techniques to build other board-game-playing programs.

Is this situation likely to change in the foreseeable future? There is currently a revival of interest in *AI general intelligence*, the attempt to build a machine that could successfully perform any intellectual task that a human being can. Is there any reason to believe that progress now will be faster than it has been since John McCarthy advocated it more than 60 years ago at the 1956 inaugural AI conference at Dartmouth? It is generally agreed that one of the key enabling technologies will be common-sense reasoning. A recent *Communications* article² argues that, while significant progress has been made in several areas of reasoning: temporal, geometric, multi-agent, and so forth, many intractable problems remain. Note also that, while successful systems, such as Watson and AlphaGo, have been applied to new areas, each of these applications is still narrow in scope. One could use a ‘Big Switch’ approach, to direct each task to the appropriate narrowly scoped system, but this approach is generally regarded as inadequate in not providing the integration of multiple cognitive processes routinely employed by humans.

Many humans tend to ascribe too much intelligence to narrowly focused AI systems.

I am not attempting to argue that AI general intelligence is, in principle, impossible. I do not believe there is anything in human cognition that is beyond scientific understanding. With such an understanding will surely come the ability to emulate it artificially. But I am not holding my breath. I have lived through too many AI hype cycles to expect the latest one to deliver something that previous cycles have failed to deliver. And I do not believe that now is the time to worry about a threat to humanity from smart machines, when there is a much more pressing problem to worry about.

That problem is that many humans tend to ascribe too much intelligence to narrowly focused AI systems. Any machine that can beat all humans at Go must surely be very intelligent, so by analogy with other world-class Go players, it must be pretty smart in other ways too, mustn’t it? No! Such misconceptions lead to false expectations that such AI systems will work correctly in areas outside their narrow expertise. This can cause problems, for example, a medical diagnosis system might recommend the wrong treatment when faced with a disease beyond its diagnostic ability, a self-driving car has already crashed when confronted by an unanticipated situation. Such erroneous behavior by dumb machines certainly presents a threat to individual humans, but not to *humanity*. To counter it, AI systems need an internal model of their scope and limitations, so that they can recognize when they are straying outside their comfort zone and warn their human users that they need human assistance or just should not be used in such a situation. We must assign a duty to AI system designers to ensure

Calendar of Events

February 4–8

PPoPP ’17: 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming
Austin, TX,
Sponsored: ACM/SIG,
Contact: Vivek Sarkar,
Email: vsarkar@rice.edu

February 6–10

WSDM 2017: 10th ACM International Conference on Web Search and Data Mining
Cambridge, U.K.,
Co-Sponsored: ACM/SIG,
Contact: Milad Shokouhi,
Email: milads@microsoft.com

February 21–22

HotMobile ’17: The 18th International Workshop on Mobile Computing Systems and Applications
Sonoma, CA,
Sponsored: ACM/SIG,
Contact: Elizabeth M. Belding,
Email: ebelding@cs.ucsb.edu

February 22–24

FPGA ’17: The 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays
Monterey, CA,
Sponsored: ACM/SIG,
Contact: Jonathan Greene,
Email: jonathan.greene@microsemi.com

February 25–March 1

CSCW ’17: Computer Supported Cooperative Work and Social Computing
Portland, OR,
Sponsored: ACM/SIG,
Contact: Steven E. Poltrock,
Email: spoltrock@gmail.com

their creations inform users of their limitations, and specifically warn users when they are asked to operate out of their scope. AI systems must have the ability to explain their reasoning in a way that users can understand and assent to. Because of their open-ended behavior, AI systems are also inherently hard to verify. We must develop software engineering techniques to address this. Since AI systems are increasingly self-improving, we must ensure these explanations, warnings, and verifications keep pace with each AI system's evolving capabilities.

The concerns of Hawkings and others were addressed in an earlier *Communications* Viewpoint by Dietterich and Horvitz.³ While downplaying these concerns, Dietterich and Horvitz also categorize the kinds of threats that AI technology *does* pose. This apparent paradox can be resolved by observing that the various threats they identify are caused by AI technology being too dumb, not too smart.

AI systems are, of course, by no means unique in having bugs or limited expertise. Any computer system deployed in a safety or security critical situation potentially poses a threat to health, privacy, finance, and other realms. That is why our field is so concerned about program correctness and the adoption of best software engineering practice. What is different about AI systems is that some people may have unrealistic expectations about the scope of their expertise, simply because they exhibit intelligence—albeit in a narrow domain.

The current focus on the very remote threat of super-human intelligence is obscuring this very real threat from sub-human intelligence.

But could such dumb machines be sufficiently dangerous to pose a threat to humanity? Yes, if, for instance, we were stupid enough to allow a dumb machine the autonomy to unleash weapons of mass destruction. We came close to such stupidity with Ronald Reagan and Edward Teller's 1983 proposal of a Strategic Defense Initiative (SDI, aka 'Star Wars').^b Satellite-based sensors would detect a Soviet ballistic missile launch and

super-powered x-ray lasers would zap these missiles from space before they got into orbit. Since this would need to be accomplished within seconds, no human could be in the loop. I was among many computer scientists who successfully argued that the most likely outcome was a false positive that would trigger the nuclear war it was designed to prevent. There were precedents from missile early-warning systems that had been triggered by, among other things, a moonrise and a flock of geese. Fortunately, in these systems a human *was* in the loop to abort any unwarranted retaliation to the falsely suspected attack. A group of us from Edinburgh met U.K. Ministry of Defence scientists, engaged with SDI, who admitted they shared our analysis. The SDI was subsequently quietly dropped by morphing it into a saner program. This is an excellent example of non-computer scientists overestimating the abilities of dumb machines. One can only hope that, like the U.K.'s MOD scientists, the developers of such weapon systems have learned the institutional lesson from this fiasco. We all also need to publicize these lessons to ensure they are widely understood. Similar problems arise in other areas too, for example, the 2010 flash crash demonstrated how vulnerable society was to the collapse of a financial system run by secret, competing and super-fast autonomous agents.

Another potential existential threat is that AI systems may automate most forms of human employment.^{7,9} If my analysis is correct then, for the foreseeable future, this automation will develop as a coalition of systems, each of which will automate only a narrowly defined task. It will be necessary for these systems to work collaboratively, with humans: orchestrating the coalition, recognizing when a system is out of its depth and dealing with these 'edge cases' interactively. The productivity of human workers will be, thereby, dramatically increased and the cost of the service provided by this multi-agent approach will be dramatically reduced, perhaps leading to an increase in the services provided. Whether this will provide both job satisfaction and a living income to all humans can currently only be an open question. It is

up to us to invent the future in which it will do, and to ensure this future is maintained as the capability and scope of AI systems increases. I do not underestimate the difficulty of achieving this. The challenges are more political and social than technical, so this is a job for the whole of society.

As AI progresses, we will see even more applications that are super-intelligent in a narrow area and incredibly dumb everywhere else. The areas of successful application will get gradually wider and the areas of dumbness narrower, but not disappear. I believe this will remain true even when we do have a deep understanding of human cognition. Maggie Boden has a nice analogy with flight. We do now understand how birds fly. In principle, we could build ever more accurate simulations of a bird, but this would incur an increasingly exorbitant cost and we already achieve satisfactory human flight by alternative means: airplanes, helicopters, paragliders, and so forth. Similarly, we will develop a zoo of highly diverse AI machines, each with a level of intelligence appropriate to its task—not a new uniform race of general-purpose, super-intelligent, humanity supplanters. □

References

1. Cellan-Jones, R. Stephen Hawking warns artificial intelligence could end mankind. BBC Interview, (Dec. 2014); <http://www.bbc.co.uk/news/technology-30290540>.
2. Davis, E. and Marcus, G. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Commun. ACM* 58, 9 (Sept. 2015), 92–103.
3. Dietterich, T.G. and Horvitz, E.J. Rise of concerns about AI: Reflections and directions. *Commun. ACM* 58, 10 (Oct. 2015), 38–40.
4. Good, I.J. Speculations concerning the first ultra-intelligent machine. *Advances in Computers* 6 (1965).
5. Kurzweil, R. *The Singularity is Near*. Penguin Group, 2005, 135–136.
6. Sloman, A. Exploring design space and niche space. In *Proceeding of the 5th Scandinavian Conference on AI*. IOS Press, Amsterdam, 1995.
7. Susskind, R. and Susskind, D. *The Future of the Professions: How Technology Will Transform the Work of Human Experts*. OUP Oxford, 2015.
8. Ulam, S. Tribute to John von Neumann. *Bulletin of the American Mathematical Society* 64, 3, part 2 (May 1958), 1–49.
9. Vardi, M.Y. The future of work: but what will humans do? *Commun. ACM* 58, 12 (Dec. 2015).
10. Warwick, K. *March of The Machines*. University of Illinois Press, 2004.

Alan Bundy (A.Bundy@ed.ac.uk) is Professor of Automated Reasoning at the School of Informatics, University of Edinburgh, Scotland.

Thanks to Stephan Schulz, Lucas Dixon, the St. Andrews University Student Debating Society, and two anonymous reviewers for feedback on earlier versions of this Viewpoint.

Copyright held by author.

^b https://en.wikipedia.org/wiki/Strategic_Defense_Initiative

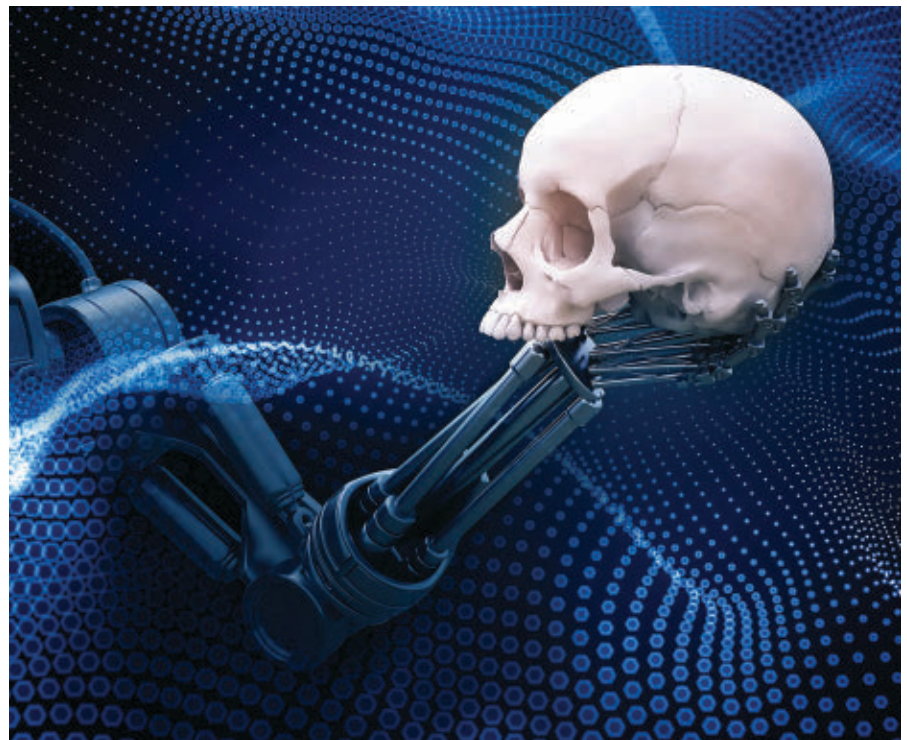
Viewpoint

AI Dangers: Imagined and Real

Arguing against the arguments for the concept of the singularity.

IN JANUARY 2015, a host of prominent figures in high tech and science and experts in artificial intelligence (AI) published a piece called “Research Priorities for Robust and Beneficial Artificial Intelligence: An Open Letter,” calling for research on the societal impacts of AI. Unfortunately, the media grossly distorted and hyped the original formulation into doomsday scenarios. Nonetheless, some thinkers do warn of serious dangers posed by AI, tacitly invoking the notion of a Technological Singularity (first suggested by Good⁸) to ground their fears. According to this idea, computational machines will improve in competence at an exponential rate. They will reach the point where they correct their own defects and program themselves to produce artificial superintelligent agents that far surpass human capabilities in virtually every cognitive domain. Such superintelligent machines could pose existential threats to humanity.

Recent techno-futurologists, such as Ray Kurzweil, posit the inevitability of superintelligent agents as the necessary result of the inexorable rate of progress in computational technology. They cite Moore’s Law for the exponential growth in the power of computer chips as the analogical basis for this claim. As the rise in the processing and storage capacity of hardware and other technologies continues, so, they maintain, will the power of AI expand, soon reaching the singularity.



These arguments for the concept of the singularity seem to us to be, at best, suspect. Moore’s Law concerns the growth of hardware processing speed. In any case, it will eventually run up against the constraints of space, time, and the laws of physics. Moreover, these arguments rely on a misplaced analogy between the exponential increase in hardware power and other technologies of recent decades and the projected rate of development in AI. Great progress is indeed being made in deep neural network learning (DL) that has produced dramatic improve-

ments in the performance of some AI systems in speech recognition, visual object recognition, object detection, and many other domains.⁴ Dramatic increase in processing power (of GPUs for example) and in the availability of large amounts of data have been the driving force behind these advances. Nevertheless, the jump from such learning to superintelligence seems to us to be more than fanciful. In the first place, almost all these advances have been in the supervised setting where there are large amounts of training data. As the leaders of DL themselves



Association for
Computing Machinery

ACM Conference Proceedings Now Available via Print-on-Demand!

*Did you know that you can
now order many popular
ACM conference proceedings
via print-on-demand?*

Institutions, libraries and individuals can choose from more than 100 titles on a continually updated list through Amazon, Barnes & Noble, Baker & Taylor, Ingram and NACSCORP: CHI, KDD, Multimedia, SIGIR, SIGCOMM, SIGCSE, SIGMOD/PODS, and many more.

**For available titles and
ordering info, visit:
librarians.acm.org/pod**



point out,⁴ this situation is the exception rather than the rule—most data is unlabeled and calls for unsupervised learning. Furthermore, these recent advances have all been in narrow specialized tasks such as image recognition, not in more general learning tasks, which require complex reasoning. Nor do these algorithms work in a recursive self-improvement loop as conceived of by Good's argument. Progress in deep learning and other areas of AI has not been exponential in any meaningful sense. It comes in irregular, and often unanticipated spurts, as is generally the case with breakthroughs in science and engineering. Unsupervised and general AI still remains a major open challenge. As pointed out in Bengio et al., "ultimately, major progress in artificial intelligence will come about through systems that combine representation learning with complex reasoning."⁴

Recent books by a mathematician¹⁰ and a philosopher⁵ have taken up variants of this view and issued their own warnings about the possible existential dangers that strong AI presents. Their main argument is more subtle. It is illustrated with a thought experiment involving the design of a machine with the narrowly defined goal of producing paper clips as efficiently as possible. Let us imagine this machine continually improves its ability to solve this narrow goal. Eventually, assuming the progress in AI continues indefinitely, the machine could set up subsidiary instrumental goals that serve its primary goal (maximizing paper clips). One of these instrumental sub-goals could conceivably be to utilize all other resources, including humans, to produce paper clips. The point of the thought experiment is to illustrate that even with a narrowly defined goal that is apparently benign, a superintelligent machine could adopt unforeseen instrumental sub-goals that are very dangerous, even to the point of posing an existential risk to humanity. Even though this is a thought experiment, both books betray a striking lack of engagement with the present state of technology in AI.

Shanahan¹¹ offers a review of the present state of AI technologies and its future possibilities in the context of the singularity. He considers various technological approaches toward superintel-

ligence. On the one hand, there is the biology-based approach of trying to understand the human brain well enough to enable whole brain emulation. Kurzweil has also promoted this perspective, sketching fantasies of nano-devices traveling through the brain to map the whole connectome. Even if it were possible to fully decipher the brain's "wiring," this does not entail that we will be able to reproduce human cognition and thought through the construction of computational models of this neural system, as Kurzweil seems to suggest, see the critique by Allen and Greaves.¹ The nematode worm has a connectome small enough to be essentially fully mapped out in 2006, but this has produced little substantive understanding of its simple brain. Recently the "Human Brain Project," a billion-euro flagship project funded by the European Commission, ran aground because of an astonishing revolt by large number of Europe's leading neuroscientists who called into question the validity of the project's basic assumptions.

Work in technology driven by AI generally seeks to solve particular tasks, rather than to model general human intelligence. It is often more efficient to perform these tasks by models that do not operate in the way that the human brain does just as we construct jets rather than machines that fly like birds. Shanahan also considers engineering AI approaches and casting them into a reinforcement learning framework. A robot is equipped with a reward function that it is programmed to optimize through interaction with the environment via a set of sensors. The agent takes actions and receives a payoff from the environment in response. It explores action strategies to maximize its payoff, and its final goal. This is perhaps the most suitable approach among today's AI technologies within which to situate Bostrom's thought experiment concerning the paper clip device whose reward function is the number of paper clips that it creates. Google's DeepMind made headlines recently by demonstrating how a combination of deep learning and reinforcement learning could be used to build a system that learns to play Atari video games, and, even more recently, that beat the world champion at the game of Go. However, for more complex tasks, Shanahan and

the Google DeepMind scientists agree that the science and technology currently associated with such an approach is in a thoroughly primitive state.

In fact much, if not all of the argument for existential risks from superintelligence seems to rest on mere logical possibility. In principle it is possible that superintelligent artificial agents could evolve, and there is no logical inconsistency in assuming they will. However, many other threats are also logically possible, but two considerations are always paramount in determining our response: a good analysis and estimate of the risk and a good understanding of the underlying natural or technological phenomena needed to formulate a response. What is the likelihood of superintelligent agents of the kind Bostrom and Haggstrom worry about? While it is difficult to compute a meaningful estimate of the probability of the singularity, the arguments here suggest to us that it is exceedingly small, at least within the foreseeable future, and this is the view of most researchers at the forefront of AI research. AI technology in its current state is also far from a mature state where credible risk assessment is possible and meaningful responses can be formulated. This can be contrasted with other areas of science and technology that pose an existential threat, for example, climate change and CRISPR gene editing. In these cases, we have a good enough understanding of the science and technology to form credible (even quantitative) threat assessment and formulate appropriate responses. Recent position papers such as Amodel et al.² ground concerns in real machine-learning research, and have initiated discussions of practical ways for engineering AI systems that operate safely and reliably.

By contrast to superintelligent agents, we are currently facing a very real and substantive threat from AI of an entirely different kind. Brynjolfsson and McAfee,⁶ and Ford⁷ show that current AI technology is automating a significant number of jobs. This trend has been increasing sharply in recent years, and it now threatens highly educated professionals from accountants to medical and legal consultants. Various reports have estimated that up to 50% of jobs in western economies like the U.S. and Sweden could be eliminated through automation over the next few decades. As Bryn-

Much, if not all of the argument for existential risks from superintelligence seems to rest on mere logical possibility.

jolfsson and McAfee note toward the end of their book, the rise of AI-driven automation will greatly exacerbate the already acute disparity in wealth between those who design, build, market, and own these systems on one hand, and the remainder of the population on the other. Reports presented at the recent WEF summit in Davos make similar predictions. Governments and public planners have not developed plausible programs for dealing with the massive social upheaval that such economic dislocation is likely to cause.

A frequently mentioned objection to this concern is that while new technologies can destroy some jobs, they also create new jobs that absorb the displaced workforce. This is how it has always been in the past. So for example, unemployed agricultural workers eventually found jobs in factories. So why should this time be different? Brynjolfsson and McAfee argue that information technologies like AI are different from previous technologies in being general-purpose technologies that have a pervasive impact across many different parts of the economy. Brynjolfsson and McAfee and Ford argue that no form of employment is immune to automation by intelligent AI systems. MIT economist David Autor points to deep and long-term structural changes in the economy as a direct result of these technologies.³

One way in which AI-powered systems can improve production and services while avoiding massive unemployment is through a partnership of people and machines, a theme running through John Markoff's book.¹⁰ He points out that the combination of humans and

machines is more powerful than either one of them alone. The strongest chess player today, for example, is neither a human, nor a computer, but a human team using computers. This is also IBM's cognitive computing vision, based on the Watson technology that defeated the human champions of "Jeopardy!" Today IBM is seeking to deploy Watson cognitive computing services in various sectors. For example, a human doctor aided by a Watson cognitive assistant would be more effective in diagnosing and treating diseases than either Watson or the doctor working separately.

While human-machine cooperation is a hopeful avenue to explore in the short to medium term, it is not clear how successful this will be, and by itself it is not an adequate solution to the social issues that AI automation poses. These constitute a major crisis of public policy. To address this crisis effectively requires that scientifically literate government planners work together with computer scientists and technologists in industry to alleviate the devastating effects of rapid technological change on the economy. The cohesion of the social order depends upon an intelligent discussion of the nature of this change, and the implementation of rational policies to maximize its general social benefit. **□**

References

1. Allen, P. and Greaves, M. The singularity isn't near. *MIT Technology Review*, 2011.
2. Amodel, D. et al. Concrete problems in AI safety. 2016. arXiv:1606.06565.
3. Autor, D.H. Why are there still so many jobs? The history and future of workplace automation. *Journal of Economic Perspectives* 29, 3 (Mar. 2015).
4. Bengio, Y., LeCun, Y., and Hinton, G. Deep Learning. *Nature* 521, 2015.
5. Bostrom, N. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, 2014.
6. Brynjolfsson, E. and McAfee, A. *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*. W.W. Norton and Co., 2016.
7. Ford, M. *Rise of the Robots: Technology and the Threat of a Jobless Future*. Basic Books, 2015.
8. Good, I.J. Speculations concerning the first ultraintelligent machine. In Franz L. Alt and Morris Rubino, Eds., *Advances in Computers*. Academic Press, 1965.
9. Haggstrom, O. *Here Be Dragons*. Oxford University Press, 2016.
10. Markoff, J. *Machines of Loving Grace: The Quest for Common Ground Between Humans and Robots*. Harper and Collins, 2015.
11. Shanahan, M. *The Technological Singularity*. MIT Press Essential Knowledge Series. MIT Press, 2015.

Devdatt Dubhashi (dubhashi@chalmers.se) is a professor in the Department of Computer Science and Engineering at Chalmers University of Technology, Sweden.

Shalom Lappin (shalom.lappin@gu.se) is a professor in the Department of Philosophy, Linguistics and Theory of Science at the University of Gothenburg, Sweden.

Copyright held by authors.

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

An apostate's opinion.

BY PAT HELLAND

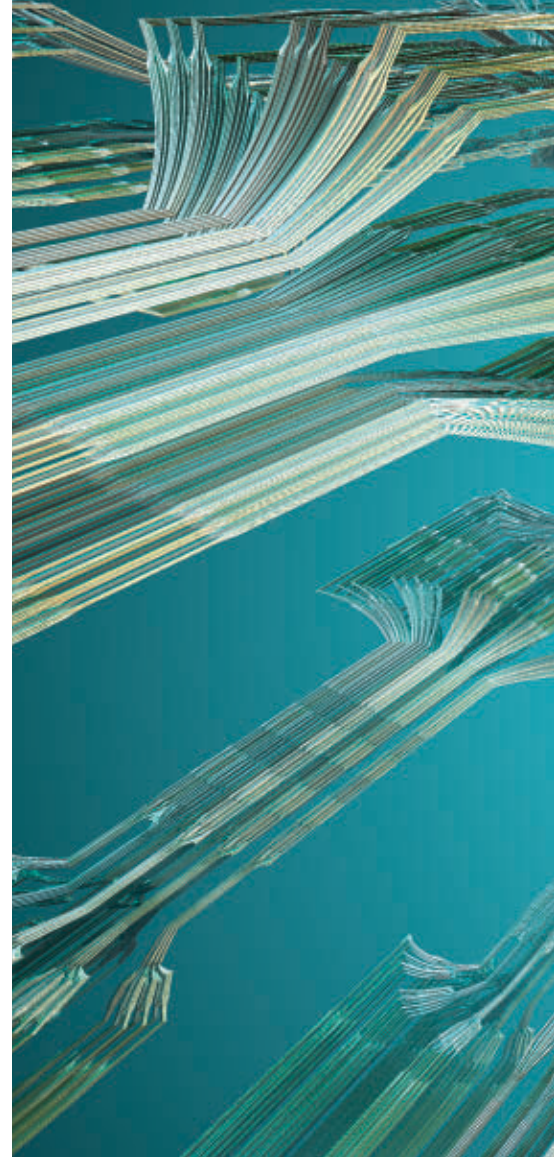
Life Beyond Distributed Transactions

TRANSACTIONS ARE AMAZINGLY powerful mechanisms, and I've spent the majority of my almost 40-year career working on them. In 1982, I first worked to provide transactions on the Tandem NonStop System. This system had a mean time between failures measured in years⁴ and included a geographically distributed two-phase commit offering excellent availability for strongly consistent transactions.

New innovations, including Google's Spanner,² offer strongly consistent transactional environments at extremely large scale with excellent availability. Building distributed transactions to support highly available applications is a great challenge that has inspired excellent innovation and amazing technology. Unfortunately, this is not broadly available to application developers.

In most distributed transaction systems, the failure of a single node causes transaction commit to stall.

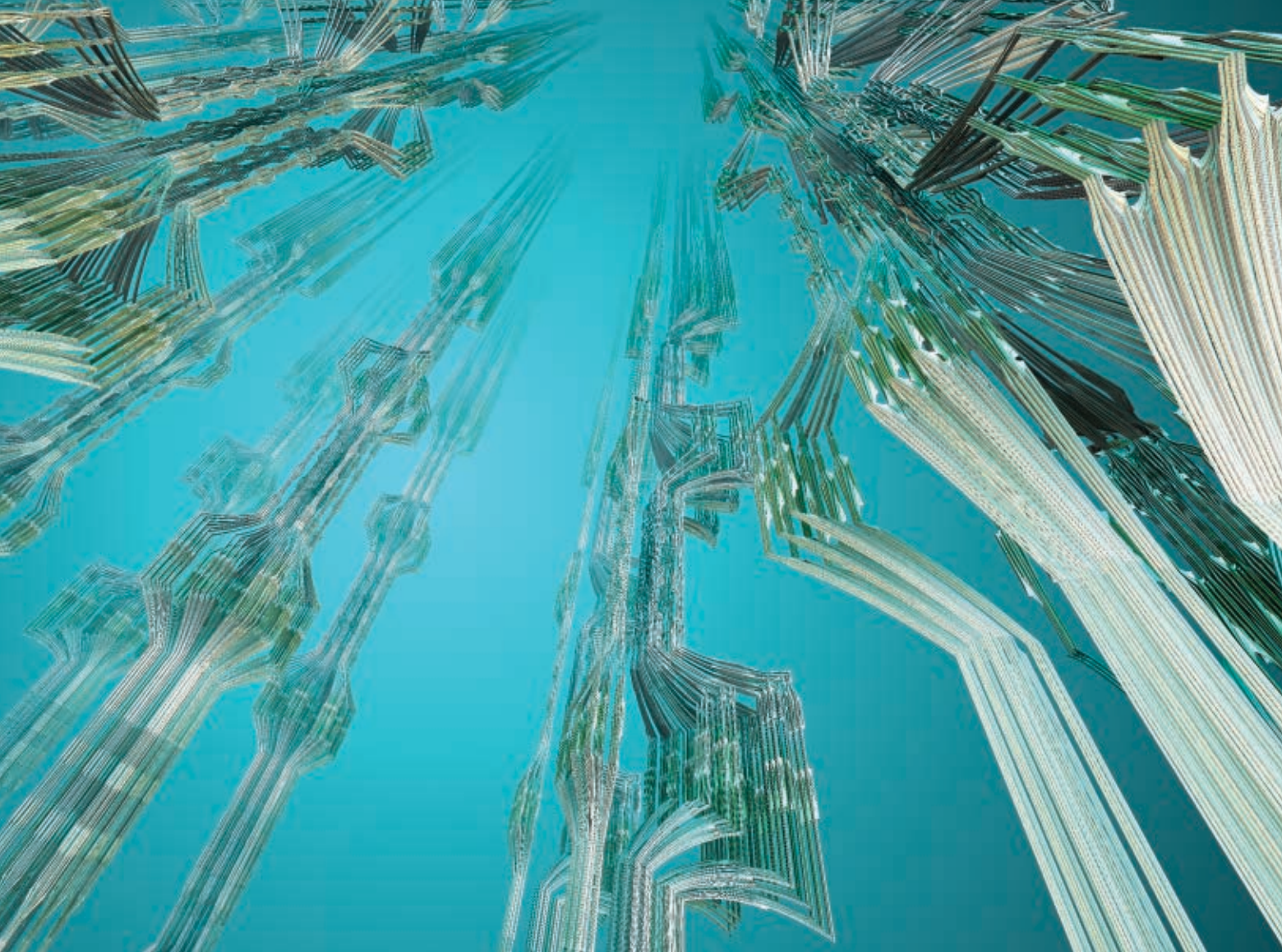
This is an updated and abbreviated version of an article by the same name first published in *Proceedings of the Conference on Innovative Database Research*, 2007.



This in turn causes the application to get wedged. In such systems, the larger it gets, the more likely the system is going to be down. When flying an airplane that needs all of its engines to work, adding an engine reduces the availability of the airplane. Running a distributed transaction system over thousands of nodes is impractical without special mechanisms to tolerate outages. When application developers build systems using non-highly available distributed transactions, the solutions are brittle and must be discarded. Natural selection kicks in...

Instead, applications are built using techniques that do not provide transactional guarantees but still meet the needs of their business.

This article explores and names some of the practical approaches used in the implementation of large-scale mission-critical applications in a world that rejects distributed transactions. Topics include the management of fine-grained pieces of application data



that may be repartitioned over time as the application grows. Design patterns support sending messages between these repartitionable pieces of data.

The goal here is to reduce the challenges faced by people handcrafting very large scalable applications. Also, by observing these design patterns, the industry can perhaps work toward the creation of platforms to make it easier to develop scalable applications. Finally, while this article targets scalable homogeneous applications, these techniques are also very useful for supporting scalable heterogeneous applications such as support for mobile devices.

Goals

This article focuses on how an application developer can build a successful scalable enterprise application when he or she has only a local database or transaction system available. Availability is not addressed, merely scale and correctness.

Discuss scalable applications.

Most designers of scalable applications understand the business requirements. The problem is that the issues, concepts, and abstractions for the interaction of transactions and scalable systems have no names and are not crisply understood. They are inconsistently applied and sometimes come back to bite the designers. One goal of this article is to launch a discussion that can increase awareness of these concepts, leading toward a common set of terms and an agreed-upon approach to scalable programs.

Think about almost-infinite scaling of applications.

The article presents an informal thought experiment on the impact of almost-infinite scaling. Let's assume the number of customers, purchasable entities, orders, shipments, healthcare patients, taxpayers, bank accounts, and all other business concepts manipulated by the app grows significantly over time. Typically, each individual

thing doesn't get much larger; there are simply more and more of them. It really doesn't matter if CPU, DRAM, storage, or some other resource gets saturated first. At some point, the increase in demand leads to spreading what used to run on a single machine over a larger number of machines. This thought experiment makes us consider tens or hundreds of thousands of machines.

Almost-infinite scaling is a loose, imprecise, and deliberately amorphous way of motivating the need to be very clear about when and where you can *know* something fits on one machine and what to do if you cannot ensure it fits on one machine. Furthermore, you want to scale almost linearly with the data and computation load. Of course, scaling at an $N\text{-log-}N$ pace with a big log would be great.

Describe a few common patterns for scalable apps. What are the impacts of almost-infinite scaling on the business logic? I am asserting that scaling im-

plies using a new abstraction called an *entity* as you write your program. An entity lives on a single machine at a time, and the application can manipulate only one entity at a time. A consequence of almost-infinite scaling is that this programmatic abstraction must be exposed to the developer of the business logic.

By naming and discussing this as-yet-unnamed concept, we can perhaps agree on a consistent programmatic approach and a consistent understanding of the issues involved in building scalable systems.

Furthermore, the use of entities has implications on the messaging patterns used to connect them. This leads to the creation of state machines that cope with the message-delivery inconsistencies foisted upon innocent application developers attempting to build scalable solutions to business problems.

Some Assumptions

Consider the following three assumptions, which are asserted and not justified. Assume these are true based on experience.

Layers of the application and scale agnosticism. Let's begin by presuming that each scalable application has

Figure 1. Two-layered application.

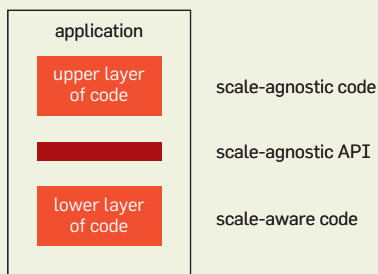
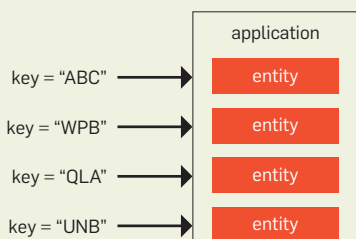


Figure 2. Data for an application comprises many entities.



at least two layers, as shown in Figure 1. These layers differ in the perception of scaling. They may have other differences, but these are not relevant to this discussion.

The lower layer of the application understands that more computers get added to make the system scale. In addition to other work, it manages the mapping of the upper layer's code to the physical machines and their locations. The lower layer is *scale-aware* in that it understands this mapping. I presume that the lower layer provides a *scale-agnostic programming abstraction* to the upper layer. There are many examples of scale-agnostic programming abstractions, including MapReduce.³

Using this scale-agnostic programming abstraction, the upper layer of the application code is written without worrying about scaling issues. By sticking to the scale-agnostic programmatic abstraction, you can write application code that is not worried about the changes happening when it is deployed over an ever-increasing load.

Over time, the lower layer of these applications may evolve to become a new platform or middleware that simplifies the creation of scale-agnostic APIs.

Transactional scopes. Lots of academic work has been done on the notion of providing strongly consistent transactions over distributed systems. This includes 2PC (two-phase commit),¹ Paxos,⁵ and recently Raft.⁶ Classic 2PC will block when a machine fails unless the coordinator and participants in the transaction are fault tolerant in their own right such as the Tandem NonStop System. Paxos and Raft do not block with node failures but do extra work coordinating much like Tandem's system.

These algorithms can be described as providing *strongly consistent transactions over distributed systems*. Their goal is to allow arbitrary atomic updates to data spread over many machines. Updates exist in a single transactional scope spanning many machines.

Unfortunately, in many circumstances this is not an option for an application developer. Applications may need to span trust boundaries, different platforms, and different operational and deployment zones. What happens when you "just say no" to distributed transactions?

Even today, 10 years after this paper was first written, real system developers rarely try to achieve strongly consistent transactions over more than just a few computers. Instead, they assume multiple separate transaction scopes. Each computer is a separate scope with local transactions inside.

Most applications use at-least-once messaging. TCP/IP is great if you are a short-lived Unix-style process, but consider the dilemma faced by an application developer whose job is to process a message and modify some durable data represented in a database. The message is consumed and not yet acknowledged. The database is updated and then the message is acknowledged. In a failure, this is restarted and the message is processed again.

The dilemma derives from the fact that the message delivery is not directly coupled to the update of the durable data other than through application action. While it is possible to couple the consumption of messages to the update of the durable data, this is not commonly available. The absence of this coupling leads to failure windows where the message is delivered more than once. Rather than lose messages, the message plumbing delivers them at least once.

A consequence of this behavior is the application must tolerate message retries and out-of-order delivery.

Opinions to be Justified

The nice thing about writing an opinion piece is that you can express wild opinions. Here are a few that this article tries to justify.

Scalable apps use uniquely identified entities. This article argues that the upper-layer code for each application must manipulate a single collection of data called an *entity*. There are no restrictions on the size of a single entity except that it must live within a single transactional scope (that is, one machine).

Each entity has a unique identifier or key, as shown in Figure 2. An entity key may be of any shape, form, or flavor. Somehow, it must uniquely identify exactly one entity and the data it contains.

There are no restrictions on the representations of the entity. It may be represented as SQL records, XML, JSON, files, or anything else. One pos-

sible representation would be a collection of SQL records, potentially across many tables, whose primary key has the entity key as its prefix.

Entities represent disjoint sets of data. Each datum resides in exactly one entity.

An application consists of many entities. For example, an order-processing application encapsulates many orders, each of which is identified by a unique Order-ID. To be a scalable order-processing application, data from one order must be disjoint from data for other orders.

Atomic transactions cannot span entities. Each computer is assumed to be a separate transactional scope. Later this article presents the argument that atomic transactions cannot span entities. The programmer must always stick to the data contained inside a single entity for each transaction.

From the programmer's perspective, the *uniquely identified entity is the transactional scope*. This concept has a powerful impact on the behavior of applications designed for scaling. One implication to be explored is that alternate indices cannot be kept transactionally consistent when designing for almost-infinite scaling.

Messages are addressed to entities. Most messaging systems do not consider the partitioning key for the data but rather target a queue that is consumed by a stateless process. Standard practice is to include some data in the message that informs the stateless application code where to get the data it needs. This is the entity key. The data for the entity is fetched from some database or other durable store by the application.

A couple of interesting trends are happening. First, the size of the *set* of entities is growing larger than will fit on a single computer. Each individual entity usually fits in one computer, but the set of them does not. Increasingly, the stateless application is routing to fetch the entity based on some partitioning scheme.

Second, the fetching and partitioning scheme is being separated into the lower layers of the application. This is deliberately isolated from the upper layers responsible for the business logic.

This pattern effectively targets the entity by routing using the entity key. Both the stateless Unix-style process

and the lower layers of the application are simply part of the implementation of the scale-agnostic API provided for the business logic. The upper-layer scale-agnostic business logic simply addresses the message to the entity key that identifies the durable state known as the entity.

Entities manage per-partner state (activities). Scale-agnostic messaging is effectively entity-to-entity messaging. The sending entity is manifest by its durable state and is identified by its entity key. It sends a message to another entity and identifies it by its entity key. The recipient entity consists of both scale-agnostic upper-layer business logic and the durable data representing its state. This is identified by its entity key.

Recall the assumption that messages are delivered at least once. The recipient entity may be assailed with redundant messages that must be ignored. In practice, messages fall into two categories: those that affect the state of the entity and those that do not. Messages that don't affect the entities state are easy—they are trivially idempotent. Messages that change the state require more care.

To ensure idempotence (that is, guarantee that the processing of retried messages is harmless), the recipient entity is typically designed to remember that the message has been processed. Once it has been successfully processed, repeated messages will typically generate another response matching the behavior of the first message.

The knowledge of the received message creates state that is wrapped up on a per-partner basis. The important observation is that the state is organized on a per-partner basis and each partner is an entity.

The term *activity* is applied to the state that manages the per-partner messaging on each side of a two-party relationship. Each activity lives in exactly one entity. An entity will have an activity for each partner entity.

In addition to managing messaging melees, activities are used to manage loosely coupled agreements. In a world where atomic transactions are not a possibility, tentative operations are used to negotiate a shared outcome. These are performed between entities and are managed by activities.

Building workflows to reach agree-

ment is fraught with challenges that are well documented elsewhere.⁷ This article does not assert that activities solve these challenges, but rather that they give a foundation for storing the state needed to solve them. Almost-infinite scaling leads to surprisingly fine-grained workflow-style solutions. The participants are entities, and each entity manages its workflow using specific knowledge about the other entities involved. That two-party knowledge maintained inside an entity is called an activity.

Examples of activities are sometimes subtle. An order application sends messages to the shipping application. It includes the shipping ID and the sending order ID. The message type may be used to stimulate the state changes in the shipping application to record that the specified order is ready to ship. Frequently, implementers don't design for retries until a bug appears. Rarely, but occasionally, the application designers think about and plan for activities.

Entities

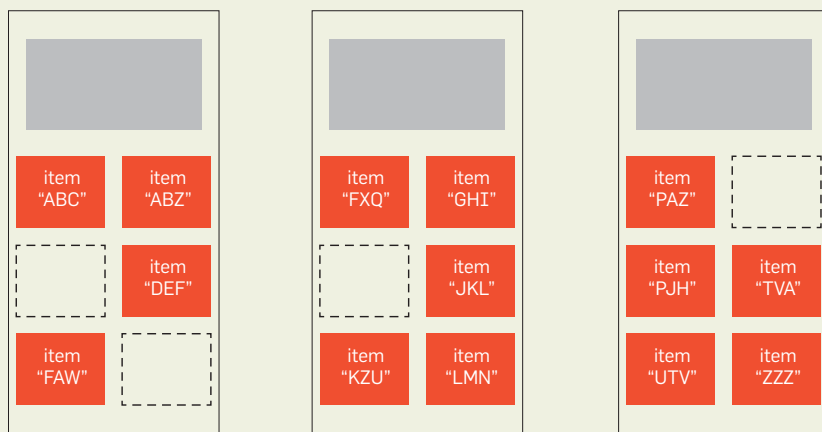
Disjoint transactional scopes. Each entity is defined as a collection of data with a unique key known to live within a single transactional scope. Atomic transactions may always be done *within a single entity*.

Uniquely keyed entities. Code for the upper layer of an application is naturally designed around collections of data with a unique key. Customer IDs, Social Security numbers, product SKUs, and other unique identifiers can be seen within applications. They are used as keys to locate the applications' data. Guarantees of transactional atomicity come only within an entity identified by a unique key.

Rep partitioning and entities. One of the assumptions previously stated is that the emerging upper layer is scale agnostic and the lower layer decides how the deployment evolves as its scale changes. The location of a specific entity is likely to change as the deployment evolves. The upper layer of the application cannot make assumptions about the location of the entity because that would not be scale agnostic.

As shown in Figure 3, entities are spread across transactional scopes using either hashing or key-range partitioning.

Figure 3. Entities spread across different transactional scopes.



Atomic transactions and entities. In scalable systems, you can't assume transactions for updates across these different entities. Each entity has a unique key, and each entity is easily placed into one transactional scope. Recall the premise that almost-infinite scaling causes the number of entities inexorably to increase, but size of the individual entity remains small enough to fit in a transactional scope (that is, one computer).

How can you know that two separate entities are guaranteed to be within the same transactional scope and, hence, atomically updatable? You know only when a single unique key unifies both. Now it is really one entity!

If hashing is used for partitioning by entity key, there's no telling when two entities with different keys land on the same box. If key-range partitioning is used for the entity keys, most of the time the adjacent key values reside on the same machine. Once in a while you will get unlucky and your neighbor will be on another machine.

A simple test case that counts on atomicity with a neighbor in a key-range partitioning will usually succeed. Later, when redeployment moves the entities across machines the latent bug emerges; the updates are no longer atomic. You can never count on different entity-key values residing in the same place.

Put more simply, the lower layer of the application will ensure that each entity key (and its entity) resides on a single machine. Different entities may be anywhere.

A scale-agnostic programming ab-

straction must have the notion of entity as the boundary of atomicity. Understanding entities, the use of the entity key, and the clear commitment to a lack of atomicity across entities are essential to scale-agnostic programming.

Large-scale applications implicitly do this in the industry today; there just isn't a name for the concept of an entity. From an upper-layer app's perspective, it must assume that the entity is the scope of transactions. Assuming more will break when the deployment changes.

Consider alternate indices. We are accustomed to the ability to address data with multiple keys or indices. For example, sometimes a customer is referenced by Social Security number, sometimes by credit-card number, and sometimes by street address. Assuming extreme amounts of scaling, these indices cannot reside on the same machine or in a single large cluster. *The data about a single customer cannot be known to reside within a single transactional scope.* The entity itself resides in a single transactional scope. The challenge is the copies of the information used to create an alternate index must be assumed to reside in a different transactional scope.

Consider guaranteeing the alternate index resides in the same transactional scope. When almost-infinite scaling kicks in, the set of entities is smeared across gigantic numbers of machines. The primary index and alternate index information must reside within the same transactional scope. The only way to ensure this is to locate the al-

ternate index using the primary index. That takes you to the same transactional scope. If you start without the primary index and have to search all of the transactional scopes, each alternate index lookup must examine an almost-infinite number of scopes as it looks for the match to the alternate key. This will eventually become untenable.

The only logical alternative is to do a two-step lookup. First, look up the alternate key, which yields the entity key. Second, access the entity using the entity key. This is very much like inside a relational database as it uses two steps to access a record via an alternate key. But the premise of almost-infinite scaling means the two indices (primary and alternate) cannot be known to reside in the same transactional scope (see Figure 4).

The scale-agnostic application program can't atomically update an entity and its alternate index. The upper-layer scale-agnostic application must be designed to understand that alternate indices may be out of sync with the entity accessed with its primary index (that is, entity key). As shown in Figure 4, different keys (primary entity key versus alternate keys) cannot be collocated or updated atomically.

What in the past has been managed automatically as alternate indices must now be managed manually by the application. Workflow-style updates via asynchronous messaging are all that are left. When you read data from alternate indices, you must understand that it is potentially out of sync with the entity itself. Alternate indices are now harder. This is a fact of life in the big cruel world of huge systems.

Messaging Across Entities

This section considers connecting independent entities using messages. It examines naming, transactions and messages, message-delivery semantics, and the impact of repartitioning entities.

Messages to communicate across entities. If you can't update the data across two entities in the same transaction, you need a mechanism to update the data in different transactions. The connection between the entities is via a message.

Asynchronous with respect to sending transactions. Since messages are across entities, the data associated with the decision to send the message is in one entity, and the destination of the message is in another entity. By definition of an entity, these entities cannot be atomically updated. Messages cannot be atomically sent and received across these different entities.

It would be horribly complex for an application developer to send a message while working on a transaction, have the message sent, and then the transaction abort. This would mean you have no memory of causing something to happen and yet it does happen. For this reason, transactional enqueueing of messages is de rigeur (see Figure 5).

If the message cannot be seen at the destination until *after* the sending transaction commits, the message is asynchronous with respect to the sending transaction. Each entity advances to a new state with a transaction. Messages are the stimuli coming from one transaction and arriving at a new entity causing transactions.

Naming the destination of messages. Consider the programming of the scale-agnostic part of an application, as one entity wants to send a message to another entity. The location of the destination entity is not known to the scale-agnostic code. The entity key is.

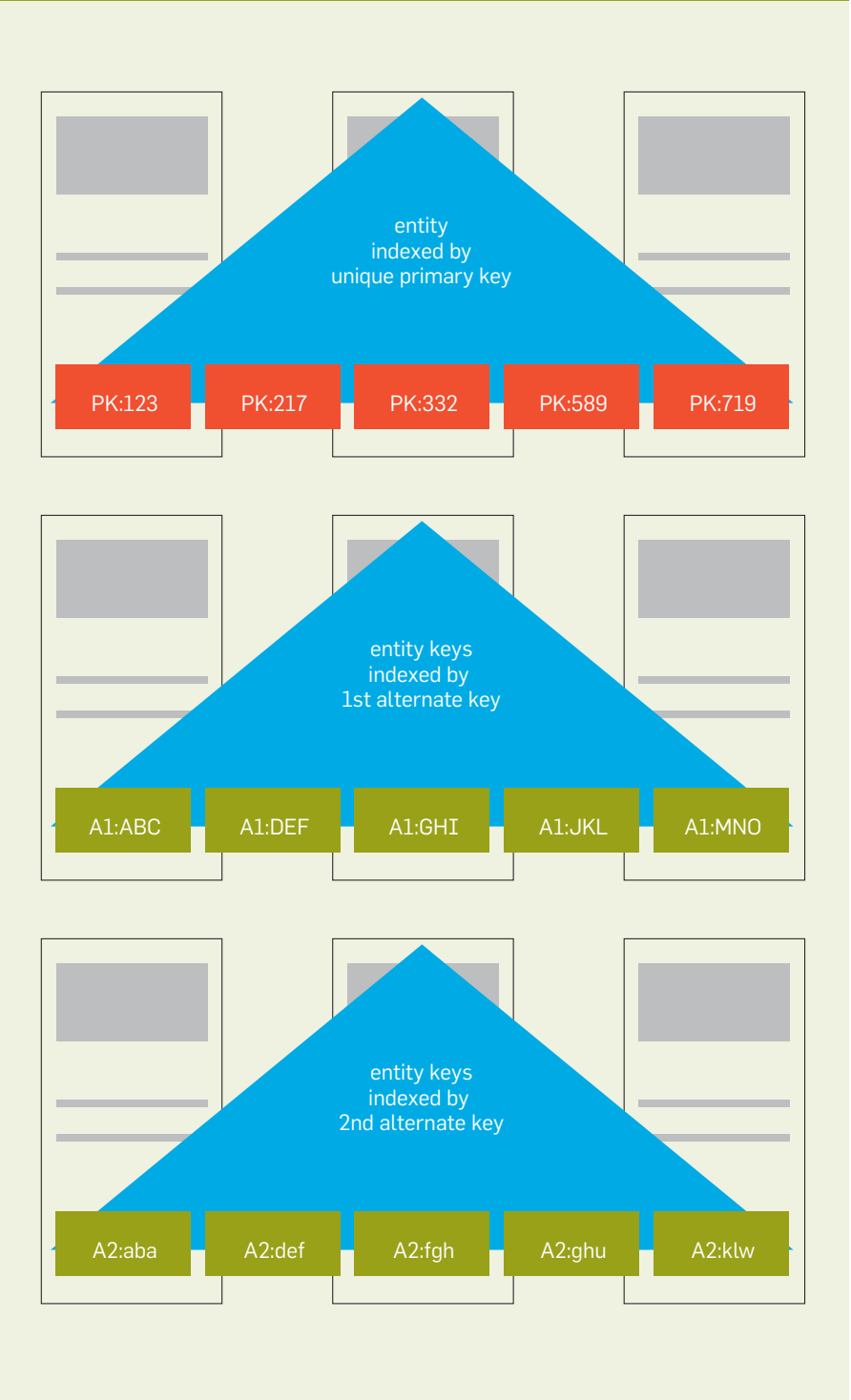
It falls on the scale-aware part of the application to correlate the entity key to the location of the entity.

Repartitioning and message delivery. When the scale-agnostic part of the application sends a message, the lower-level scale-aware portion hunts down the destination and delivers the message at least once.

As the system scales, entities move. This is commonly called *repartitioning*. The location for the entity and, hence, the destination for the message may be in flux. Sometimes messages will chase to the old location only to find out the pesky entity has been sent elsewhere. Now the message will have to follow.

As entities move, the clarity of a first-in, first-out queue between the sender and the destination is occasionally disrupted. Messages are repeated. Later messages arrive before earlier ones. Life gets messier.

Figure 4. Under scale, alternative index entries land elsewhere.

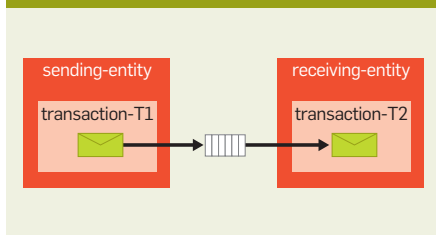


For these reasons, scale-agnostic applications are evolving to support idempotent processing of all application-visible messaging. This implies reordering in message delivery, too.

Activities: Coping with Messy Messages

This section discusses ways of coping with the challenges of message retries

Figure 5. Messages flow from one entity to another.



and reordering. It introduces the notion of an activity as the local information needed to manage a relationship with a partner entity.

Retries and idempotence. Since any message may be delivered multiple times, the application needs a discipline to cope with repeated messages. While it is possible to build low-level support for the elimination of duplicate messages, in an almost-infinite scaling environment the low-level support would need to know about entities. The knowledge of which messages have been delivered to the entity must travel with the entity when it moves because of repartitioning. In practice, the low-level management of this knowledge rarely occurs; messages may be delivered more than once.

Typically, the scale-agnostic (higher-level) portion of the application must implement mechanisms to ensure that the incoming message is idempotent. This is not essential to the nature of the problem. Duplicate elimination could

certainly be built into the scale-aware parts of the application. This is not yet available. Hence, let's consider what the poor developer of the scale-agnostic application must implement.

Defining idempotence of substantive behavior. The processing of a message is idempotent if a subsequent execution of the processing does not perform a *substantive change* to the entity. This is an amorphous definition that leaves open to the application the specification of what is and what is not substantive.

If a message does not change the invoked entity but only reads information, its processing is idempotent. This is true even if a log record describing the read is written. The log record is not substantive to the behavior of the entity. The definition of what is and what is not substantive is application specific.

Natural idempotence. To accomplish idempotence, it is essential that the message does not cause substantive side effects. Some messages provoke no substantive work any time they are processed. These are *naturally* idempotent.

A message that only reads some data from an entity is naturally idempotent. What if the processing of a message does change the entity but not in a substantive way? Those, too, would be naturally idempotent.

Now it gets more difficult. The work implied by some messages actually cause substantive changes. These messages are not naturally idempotent.

The application must include mechanisms to ensure that these, too, are idempotent. This means remembering in some fashion that the message has been processed so that subsequent attempts make no substantive change.

Remembering messages as state. To ensure the idempotent processing of messages that are not naturally idempotent, the entity must remember they have been processed. This knowledge is state. The state accumulates as messages are processed.

In addition, if a reply is required, the same reply must be returned. After all, you don't know if the original sender has received the reply.

Manage state for each partner. To track relationships and the messages received, each entity within the scale-agnostic application must somehow remember state information about its partners. It must capture this state on a partner-by-partner basis. Let's name this state an *activity* (see Figure 6). Each entity may have many activities if it interacts with many other entities. Activities track the interactions with each partner.

Each entity consists of a set of activities and, perhaps, some other data that spans the activities.

Consider the processing of an order consisting of many items for purchase. Reserving inventory for shipment of each separate item will be a separate activity. There will be an entity for the order and separate entities for each item managed by the warehouse. Transactions cannot be assumed across these entities.

Within the order, each inventory item will be separately managed. The messaging protocol must be separately managed. The per-inventory-item data contained within the order entity is an activity. While it is not named as such, this pattern frequently exists in large-scale apps.

In an almost-ininitely scaled application, you need to be very clear about relationships. You can't just do a query to figure out what is related. Everything must be formally knit together using a web of two-party relationships. The knitting is done with the entity keys. Because the partner is some distance away, you have to formally manage your understanding of the partner's state

Figure 6. Activities are what an entity remembers.

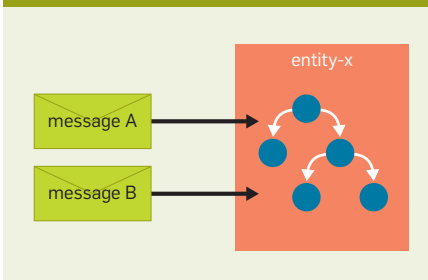
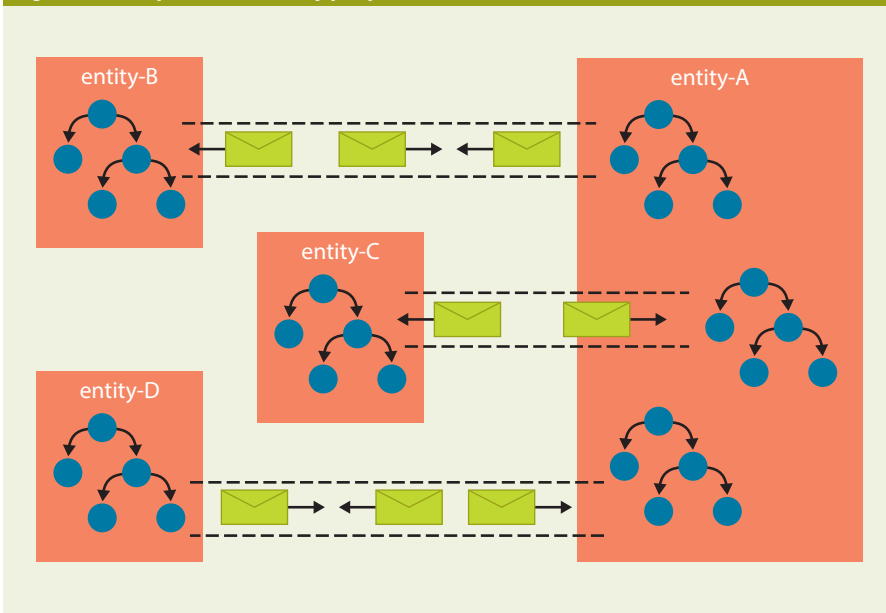


Figure 7. Example of one activity per partner.



as new knowledge when the partner arrives. The local information known about a distant partner is referred to as an activity (see Figure 7).

Ensuring at-most-once acceptance via activities. Processing messages that are not naturally idempotent requires ensuring that each message is processed at most once (that is, the substantive impact of the message must happen at most once). To do this, some unique characteristic of the message must be remembered to ensure it will not be processed more than once.

The entity must durably remember the transition from a message being OK to process into the state where the message will not have substantive impact.


Typically, an entity uses its activities to implement this state management on a partner-by-partner basis. This is essential because sometimes an entity supports many different partners, and each will pass through a pattern of messages associated with that relationship. The per-partner collection of state makes that possible.

Activities: Coping Without Atomicity


Managing distributed agreement is hard work. In an almost-infinitely scalable environment, the representation of uncertainty must be done in a fine-grained fashion that is oriented around per-partner relationships. This data is managed within entities using the notion of an activity.

Uncertainty at a distance. The absence of distributed transactions means the acceptance of uncertainty when attempting to come to decisions across different entities. It is unavoidable that decisions across distributed systems involve accepting uncertainty for a while. When distributed transactions can be used, that uncertainty is manifest in the locks held on data and is managed by the transaction manager.

In a system that cannot count on distributed transactions, the management of uncertainty must be implemented in the business logic. The uncertainty of the outcome is held in the business semantics rather than in the record lock. This is simply workflow. It's not magic. You can't



Almost-infinite scaling is a loose, imprecise, and deliberately amorphous way of motivating the need to be very clear about when and where you can know something fits on one machine and what to do if you cannot ensure it.



use distributed transactions, so you use workflow.

The assumptions that led to entities and messages now lead to the conclusion that the scale-agnostic application must manage uncertainty itself using workflow. This is needed to reach agreement across multiple entities.

Think about the style of interactions common across businesses. Contracts between businesses include time commitments, cancellation clauses, reserved resources, and much more. Uncertainty is wrapped up in the behavior of the business functionality. While more complicated to implement than using distributed transactions, it is how the real world works...

Again, this is simply an argument for workflow.

Activities and the management of uncertainty. Entities sometimes accept uncertainty as they interact with other entities. This uncertainty must be managed on a partner-by-partner basis and can be visualized as being reified in the activity state for the partner.

Many times, uncertainty is represented by relationship. It is necessary to track it by partner. The activity tracks each partner as it advances into a new state.

Performing tentative business operations. To reach an agreement across entities, one entity asks another to accept uncertainty. One entity sends another a request that may be canceled later. This is called a *tentative operation*. At the end of this step, one entity agrees to abide by the wishes of the other.

Tentative operations, confirmation, and cancellation. Essential to a tentative operation is the right to cancel. If the requesting entity decides not to move forward, it issues a *cancellation operation*. If it decides to move ahead, it issues a *confirmation operation*.

If an ordering system reserves inventory from a warehouse, the warehouse allocates the inventory without knowing if it will be used. That is accepting uncertainty. Later, the warehouse finds out if the reserved inventory will be needed. This resolves the uncertainty.

The warehouse inventory manager must keep relationship data for each

order encumbering its items. As it connects items and orders, these will be organized by item. Each item keeps information about outstanding orders for that item. Each activity within the item (one per order) manages the uncertainty of the order.

When an entity agrees to perform a tentative operation, it agrees to let another entity decide the outcome. It accepts uncertainty, which adds to its confusion. As confirmations and cancellations arrive, the uncertainty decreases, reducing confusion. It is normal to proceed through life with ever increasing and decreasing uncertainty as old problems get resolved and new ones arrive in your lap.

Again, this is simply workflow, but it is fine-grained workflow with entities as the participants.

Uncertainty and almost-infinite scaling. The management of uncertainty usually revolves around two-party agreements. There may be multiple two-party agreements. These are knit together as a web of fine-grained two-party agreements using entity keys as the links and activities to track the known state of a distant partner.

Consider a house purchase and the relationships with the escrow company. The buyer enters into an agreement of trust with the escrow company, as do the seller, mortgage company, and all other parties involved in the transaction.

When you go to sign papers to buy a house, you do not know the outcome of the deal. You accept that, until escrow closes, you are uncertain. The only party with control over the decision-making is the escrow company.

This is a hub-and-spoke collection of two-party relationships that are used to get a large set of parties to agree without use of distributed transactions.

When considering almost-infinite scaling, it is interesting to think about two-party relationships. By building up from two-party tentative/cancel/confirm (just like traditional workflow), you can see the basis for achieving distributed agreement. Just as in the escrow company, many entities may participate in an agreement through composition.

Because the relationships are

two-party, the simple concept of an activity as “stuff I remember about that partner” becomes a basis for managing enormous systems—even when the data is stored in entities and you don’t know where the entity lives. You must assume it is far away. Still, it can be programmed in a scale-agnostic way.

Real-world almost-infinite scale applications would love the luxury of a global transactional scope. Unfortunately, this is not readily available to most of us without introducing fragility when a system fails.

Instead, the management of the uncertainty of the tentative work is passed off into the hands of the developer of the scale-agnostic application. It must be handled as reserved inventory, allocations against credit lines, and other application-specific concepts.

Conclusion

As usual, the computer industry is in flux.

Today, new design pressures are being foisted onto programmers who simply want to solve business problems. Their realities are taking them into a world of almost-infinite scaling and forcing them into design problems largely unrelated to the real business at hand. This is as true today, as it was when this article was first published in 2007.

Unfortunately, programmers striving to solve business goals such as e-commerce, supply-chain-management, financial, and health-care applications increasingly need to think about scaling without distributed transactions. Most developers simply don’t have access to robust systems offering scalable distributed transactions.

We are at a juncture where the patterns for building these applications can be seen, but no one is yet applying these patterns consistently. This article argues that these nascent patterns can be applied more consistently in the handcrafted development of applications designed for almost-infinite scaling.

This article has introduced and named a couple of formalisms emerging in large-scale applications:

► *Entities* are collections of named (keyed) data that may be atomically updated within the entity but never atom-

ically updated across entities.

► *Activities* consist of the collection of state within the entities used to manage messaging relationships with a single partner entity.

Workflow to reach decisions functions within activities within entities. It is the fine-grained nature of workflow that is surprising when looking at almost-infinite scaling.

Many applications are implicitly being designed with both entities and activities today. They are simply not formalized, nor are they consistently used. Where the use is inconsistent, bugs are found and eventually patched. By discussing and consistently using these patterns, better large-scale applications can be built and, as an industry, we can get closer to building solutions that allow business-logic programmers to concentrate on the business problems rather than the problems of scale. □

Related articles on queue.acm.org

A Conversation with Bruce Lindsay

<http://queue.acm.org/detail.cfm?id=1036486>

Condos and Clouds

By Pat Helland

<http://queue.acm.org/detail.cfm?id=2398392>

Microsoft's Protocol Documentation Program: Interoperability Testing at Scale

A discussion with Nico Kicillof, Wolfgang

Grieskamp, and Bob Binder

<http://queue.acm.org/detail.cfm?id=1996412>

References

- Bernstein, P.A., Hadzilacos, V. and Goodman, N. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, Boston, MA, 1987.
- Corbett, J.C. et al. Spanner: Google's globally distributed database. In *Proceedings of the 10th Usenix Symposium on Operating Systems Design and Implementation*, 2012.
- Dean, J. and Ghemawat, S. MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation*, 2004.
- Gray, J. A census of Tandem Systems availability between 1985 and 1990. *IEEE Transactions on Reliability* 39, 4 (1990), 409–418.
- Lampert, L. The part-time parliament. *ACM Trans. Computer Systems* 16, 2 (1998), 133–169.
- Ongaro, D. and Ousterhout, J. In search of an understandable consensus algorithm (extended version), 2014; <https://raft.github.io/raft.pdf>.
- Wachter, H. and Reuter, A. The ConTract Model. *Database Transaction Models for Advanced Applications*. 0 219–263. Morgan Kaufmann, San Francisco, CA, 1992, 219–263.

Pat Helland has been implementing transaction systems, databases, application platforms, distributed systems, fault-tolerant systems, and messaging systems since 1978. He currently works at Salesforce.

Copyright held by author.
Publication rights licensed to ACM \$15.00

**Anyone can use a load balancer.
Using it properly is much more difficult.**

BY THOMAS A. LIMONCELLI

Are You Load Balancing Wrong?

A READER CONTACTED me recently to ask if it is better to use a load balancer to add capacity or to make a service more resilient to failure. The answer is: both are appropriate uses of a load balancer. The problem, however, is that most people who use load balancers are doing it wrong.

In today's Web-centric, service-centric environments the use of load balancers is widespread. I assert, however, that most of the time they are used incorrectly. To understand the problem, we first need to discuss a little about load balancers in general. Then we can look at the problem and solutions.

A load balancer receives requests and distributes them to two or more machines. These machines are called replicas, as they provide the same service. For the sake of simplicity, assume these are HTTP requests from Web browsers, but load balancers can also be used with HTTPS requests, DNS queries, SMTP (email) connections, and many other protocols. Most modern applications are engineered to work behind a load balancer.

There are two primary ways to use load balancers: to increase capacity and to improve resiliency.

Using a load balancer to increase capacity is very simple. If one replica is not powerful enough to handle the entire incoming workload, a load balancer can be used to distribute the workload among multiple replicas.

Suppose a single replica can handle 100QPS (queries per second). As long as fewer than 100QPS are arriving, it should run fine. If more than 100QPS arrive, then the replica becomes overloaded, rejects requests, or crashes. None of these is a happy situation.

If there are two machines behind a load balancer configured as replicas, then capacity is 200QPS; three replicas would provide 300QPS of capacity, and so on. As more capacity is needed, more replicas can be added. This is *horizontal scaling*.

Load balancers can also be used to improve resiliency. Resilience means the ability to survive a failure.

Individual machines fail, but the system should continue to provide service. All machines eventually fail—that's physics. Even if a replica had near-perfect uptime, you would still need resiliency mechanisms because of other externalities such as software upgrades or the need to physically move a machine.

A load balancer can be used to achieve resiliency by leaving enough spare capacity that a single replica can fail and the remaining replicas can handle the incoming requests.


Continuing the example, suppose four replicas have been deployed to achieve 400QPS of capacity. If you are currently receiving 300QPS, each replica will receive approximately 75QPS (one-quarter of the workload). What will happen if a single replica fails? The load balancer will quickly see the outage and shift traffic such that each replica receives about 100QPS. That means each replica is running at maximum capacity. That's cutting it close, but it is acceptable.

What if the system had been receiving 400QPS? Under normal operation, each of the four replicas would receive approximately 100QPS. If a single replica died, however, the remaining replicas would receive approximately 133QPS each. Since each replica can process about 100QPS, this means each one of them is overloaded by a third. The system might slow to a crawl and become unusable. It might crash.


The determining factor in how the load balancer was used is whether or not the arriving workload was above or below 300QPS. If 300 or fewer QPS were arriving, this would be a load balancer used for resiliency. If 301 or more QPS were arriving, this would be a load balancer for increased capacity.

The difference between using a load balancer to increase capacity or improve resiliency is an operational difference, not a configuration difference. Both use cases configure the hardware and network (or virtual hardware and virtual network) the same, and configure the load balancer with the same settings.

The term *N+1 redundancy* refers to a system that is configured such that if a single replica dies, enough capacity is left over in the remaining *N* replicas for the system to work properly. A system



There are two primary ways to use load balancers: to increase capacity and to improve resiliency.



is *N+0* if there is no spare capacity. A system can also be designed to be *N+2* redundant, which would permit the system to survive two dead replicas, and so on.

Three Ways to Do It Wrong

Now that we understand two different ways a load balancer can be used, let's examine how most teams fail.

Level 1: The Team Disagrees

Ask members of the team whether the load balancer is being used to add capacity or improve resiliency. If different people on the team give different answers, you're load balancing wrong.

If the team disagrees, then different members of the team will be making different engineering decisions. At best, this leads to confusion. At worst, it leads to suffering.

You would be surprised at how many teams are at this level.

Level 2: Capacity Undefined

Another likely mistake is not agreeing how to measure the capacity of the system. Without this definition, you do not know if this system is *N+0* or *N+1*. In other words, you might have agreement that the load balancing is for capacity or resilience, but you do not know whether or not you are using it that way.

To know for sure, you have to know the actual capacity of each replica. In an ideal world, you would know how many QPS each replica can handle. The math to calculate the *N+1* threshold (or high-water mark) would be simple arithmetic. Sadly, the world is not so simple.

You can't simply look at the source code and know how much time and resources each request will require and determine the capacity of a replica. Even if you did know the theoretical capacity of a replica, you would need to verify it experimentally. We are scientists, not barbarians!

Capacity is best determined by benchmarks. Queries are generated and sent to the system at different rates, with the response times measured. Suppose you consider a 200ms response time to be sufficient. You can start by generating queries at 50 per second and slowly increase the rate until the system is overloaded and responds slower than 200ms. The last QPS rate that resulted

in sufficiently fast response times determines the capacity of the replica.

How do you quantify response time when measuring thousands or millions of queries? Not all queries run in the same amount of time. You can't take the average, as a single long-running request could result in a misleading statistic. Averages also obscure bimodal distributions. (For more on this, see chapter 17, Monitoring Architecture and Practice, of *The Practice of Cloud System Administration, Volume 2*, by T. Limoncelli, S.R. Chalup, and C.J. Hogan; Addison-Wesley, 2015).

Since a simple average is insufficient, most sites use a percentile. For example, the requirement might be that the 90th percentile response time must be 200ms or better. This is a very easy way to toss out the most extreme outliers. Many sites are starting to use MAD (median absolute deviation), which is explained in a 2015 paper by David Goldberg and Yinan Shan, "The Importance of Features for Statistical Anomaly Detection" (<https://www.usenix.org/system/files/conference/hotcloud15/hotcloud15-goldberg.pdf>).

Generating synthetic queries to use in such benchmarks is another challenge. Not all queries take the same amount of time. There are short and long requests. A replica that can handle 100QPS might actually handle 80 long queries and 120 short queries. The benchmark must use a mix that reflects the real world.

If all queries are read-only or do not mutate the system, you can simply record an hour's worth of actual queries and replay them during the benchmark. At a previous employer, we had a dataset of 11 billion search queries used for benchmarking our service. We would send the first 1 billion queries to the system to warm up the cache. We recorded measurements during the remaining queries to gauge performance.

Not all workloads are read-only. If a mixture of read and write queries is required, the benchmark dataset and process is much more complex. It is important that the mixture of read and write queries reflects real-world scenarios.

Sadly, the mix of query types can change over time as a result of the introduction of new features or unanticipated changes in user-access patterns. A system that was capable of 200QPS today

may be rated at 50QPS tomorrow when an old feature gains new popularity.

Software performance can change with every release. Each release should be benchmarked to verify that capacity assumptions have not changed.

If this benchmarking is done manually, there's a good chance it will be done only on major releases or rarely. If the benchmarking is automated, then it can be integrated into your continuous integration (CI) system. It should fail any release that is significantly slower than the release running in production. Such automation not only improves engineering productivity because it eliminates the manual task, but also boosts engineering productivity because you immediately know the exact change that caused the regression. If the benchmarks are done occasionally, then finding a performance regression involves hours or days of searching for which change caused the problem.

Ideally, the benchmarks are validated by also measuring live performance in production. The two statistics should match up. If they don't, you must true-up the benchmarks.

Another reason why benchmarks are so complicated is caches. Caches have unexpected side effects. For example, intuitively you would expect that a system should get faster as replicas are added. Many hands make light work. Some applications get slower with more replicas, however, because cache utilization goes down. If a replica has a local cache, it is more likely to have a cache hit if the replica is highly utilized.

Level 3: Definition But No Monitoring

Another mistake a team is likely to make is to have all these definitions agreed upon, but no monitoring to detect whether or not you are in compliance.

Suppose the team has determined that the load balancer is for improving both capacity and resilience, they have defined an algorithm for measuring the capacity of a replica, and they have done the benchmarks to ascertain the capacity of each replica.

The next step is to monitor the system to determine whether the system is $N+1$ or whatever the desired state is.

The system should not only monitor the utilization and alert the operations team when the system is out of compli-

ance, but also alert the team when the system is nearing that state. Ideally, if it takes T minutes to add capacity, the system must send the alert at least T minutes before that capacity is needed.

Cloud-computing systems such as Amazon Web Services (AWS) have systems that can add more capacity on demand. If you run your own hardware, provisioning new capacity may take weeks or months. If adding capacity always requires a visit to the CFO to sign a purchase order, you are not living in the dynamic, fast-paced, high-tech world you think you are.

Summary

Anyone can use a load balancer. Using it properly is much more difficult. Some questions to ask:

1. Is this load balancer used to increase capacity ($N+0$) or to improve resiliency ($N+1$)?
2. How do you measure the capacity of each replica? How do you create benchmark input? How do you process the benchmark results to arrive at the threshold between good and bad?
3. Are you monitoring whether you are compliant with your $N+M$ configuration? Are you alerting in a way that provides enough time to add capacity so that you stay compliant?

If the answer to any of these questions is "I don't know" or "No," then you're doing it wrong. ■

Related articles

The Tail at Scale

Jeffrey Dean, Luiz André Barroso
Communications of the ACM 56, 2
(Feb. 2013), 74–80.

<http://cacm.acm.org/magazines/2013/2/160173-the-tail-at-scale/abstract>

Resilience Engineering: Learning to Embrace Failure

A discussion with Jesse Robbins, Kripa Krishnan, John Allspaw, and Tom Limoncelli
<http://queue.acm.org/detail.cfm?id=2371297>

The Pathologies of Big Data

Adam Jacobs
<http://queue.acm.org/detail.cfm?id=1563874>

Thomas A. Limoncelli is a site reliability engineer at Stack Overflow Inc. in New York City. His books include *The Practice of Cloud Administration* (<http://the-cloud-book.com>), *The Practice of System and Network Administration* (<http://the-sysadmin-book.com>), and *Time Management for System Administrators*. He blogs at EverythingSysadmin.com and tweets at @YesThatTom.

Copyright held by author.
Publication rights license to ACM. \$15.00

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

Measuring bottleneck bandwidth and round-trip propagation time.

BY NEAL CARDWELL, YUCHUNG CHENG, C. STEPHEN GUNN,
SOHEIL HASSAS YEGANEH, AND VAN JACOBSON

BBR: Congestion-Based Congestion Control

BY ALL ACCOUNTS, today's Internet is not moving data as well as it should. Most of the world's cellular users experience delays of seconds to minutes; public Wi-Fi in airports and conference venues is often worse. Physics and climate researchers need to exchange petabytes of data with global collaborators but find their carefully engineered multi-Gbps infrastructure often delivers at only a few Mbps over intercontinental distances.⁶

These problems result from a design choice made when TCP congestion control was created in the 1980s—interpreting packet loss as “congestion.”¹³ This equivalence was true at the time but was because of technology limitations, not first principles. As NICs (network interface controllers) evolved from Mbps to Gbps and memory chips from KB to GB, the relationship between packet loss and congestion became more tenuous.

Today TCP's loss-based congestion control—even with the current best of breed, CUBIC¹¹—is the primary

cause of these problems. When bottleneck buffers are large, loss-based congestion control keeps them full, causing bufferbloat. When bottleneck buffers are small, loss-based congestion control misinterprets loss as a signal of congestion, leading to low throughput. Fixing these problems requires an alternative to loss-based congestion control. Finding this alternative requires an understanding of where and how network congestion originates.

Congestion and Bottlenecks

At any time, a (full-duplex) TCP connection has exactly one slowest link or *bottleneck* in each direction. The bottleneck is important because:

- ▶ It determines the connection's maximum data-delivery rate. This is a general property of incompressible flow (for example, picture a six-lane freeway at rush hour where an accident has reduced one short section to a single lane. The traffic upstream of the accident moves no faster than the traffic through that lane).

- ▶ It is where persistent queues form. Queues shrink only when a link's departure rate exceeds its arrival rate. For a connection running at maximum delivery rate, all links upstream of the bottleneck have a faster departure rate so their queues migrate to the bottleneck.

Regardless of how many links a connection traverses or what their individual speeds are, from TCP's viewpoint an arbitrarily complex path behaves as a single link with the same RTT (round-trip time) and bottleneck rate. Two physical constraints, $RTprop$ (round-trip propagation time) and $BtlBw$ (bottleneck bandwidth), bound transport performance. (If the network path were a physical pipe, $RTprop$ would be its length and $BtlBw$ its minimum diameter.)

Figure 1 shows RTT and delivery rate variation with the amount of data *in flight* (data sent but not yet acknowledged). Blue lines show the $RTprop$ constraint, green lines the $BtlBw$ constraint, and red lines the bottleneck buffer. Operation in the shaded regions is not possible since it would vio-



late at least one constraint. Transitions between constraints result in three different regions (app-limited, bandwidth-limited, and buffer-limited) with qualitatively different behavior.

When there isn't enough data in flight to fill the pipe, $RTprop$ determines behavior; otherwise, $BtlBw$ dominates. Constraint lines intersect at $inflight = BtlBw \times RTprop$, a.k.a. the pipe's BDP (bandwidth-delay product). Since the pipe is full past this point, the $inflight$ -BDP excess creates a queue

at the bottleneck, which results in the linear dependence of RTT on $inflight$ data shown in the upper graph. Packets are dropped when the excess exceeds the buffer capacity. *Congestion* is just sustained operation to the right of the BDP line, and *congestion control* is some scheme to bound how far to the right a connection operates on average.

Loss-based congestion control operates at the right edge of the bandwidth-limited region, delivering full bottleneck bandwidth at the cost of high

delay and frequent packet loss. When memory was expensive buffer sizes were only slightly larger than the BDP, which minimized loss-based congestion control's excess delay. Subsequent memory price decreases resulted in buffers orders of magnitude larger than ISP link BDPs, and the resulting bufferbloat yielded RTTs of seconds instead of milliseconds.⁹

The left edge of the bandwidth-limited region is a better operating point than the right. In 1979, Leonard Klein-

rock¹⁶ showed this operating point was optimal, maximizing delivered bandwidth while minimizing delay and loss, both for individual connections and for the network as a whole⁸. Unfortunately, around the same time Jeffrey M. Jaffe¹⁴ proved it was impossible to create a distributed algorithm that converged to this operating point. This result changed the direction of research from finding a distributed algorithm that achieved Kleinrock's optimal operating point to investigating different approaches to congestion control.

Our group at Google spends hours each day examining TCP packet header captures from all over the world, making sense of behavior anomalies and pathologies. Our usual first step is finding the essential path characteris-

tics, $RTprop$ and $BtlBw$. That these can be inferred from traces suggests that Jaffe's result might not be as limiting as it once appeared. His result rests on fundamental measurement ambiguities (For example, whether a measured RTT increase is caused by a path-length change, bottleneck bandwidth decrease, or queuing delay increase from another connection's traffic). Although it is impossible to disambiguate any single measurement, a connection's behavior over time tells a clearer story, suggesting the possibility of measurement strategies designed to resolve ambiguity.

Combining these measurements with a robust servo loop using recent control systems advances¹² could result in a distributed congestion-control

protocol that reacts to actual congestion, not packet loss or transient queue delay, and converges with high probability to Kleinrock's optimal operating point. Thus began our three-year quest to create a congestion control based on measuring the two parameters that characterize a path: bottleneck bandwidth and round-trip propagation time, or BBR.

Characterizing the Bottleneck

A connection runs with the highest throughput and lowest delay when (rate balance) the bottleneck packet arrival rate equals $BtlBw$ and (full pipe) the total data in flight is equal to the BDP ($= BtlBw \times RTprop$).

The first condition guarantees that the bottleneck can run at 100% utilization. The second guarantees there is enough data to prevent bottleneck starvation but not overflow the pipe. The rate balance condition alone *does not* ensure there is no queue, only that it cannot change size (for example, if a connection starts by sending its 10-packet Initial Window into a five-packet BDP, then runs at exactly the bottleneck rate, five of the 10 initial packets fill the pipe so the excess forms a standing queue at the bottleneck that cannot dissipate). Similarly, the full pipe condition does not guarantee there is no queue (for example, a connection sending a BDP in BDP/2 bursts gets full bottleneck utilization, but with an average queue of BDP/4). The only way to minimize the queue at the bottleneck and all along the path is to meet both conditions simultaneously.

$BtlBw$ and $RTprop$ vary over the life of a connection, so they must be continuously estimated. TCP currently tracks RTT (the time interval from sending a data packet until it is acknowledged) since it is required for loss detection. At any time t ,

$$RTT_t = RTprop_t + \eta_t$$

where $\eta \geq 0$ represents the "noise" introduced by queues along the path, the receiver's delayed ack strategy, ack aggregation, and so on. $RTprop$ is a physical property of the connection's path and changes only when the path changes. Since path changes happen on time scales $\gg RTprop$, an unbiased, efficient estimator at time T is

Figure 1. Delivery rate and round-trip time vs. inflight.

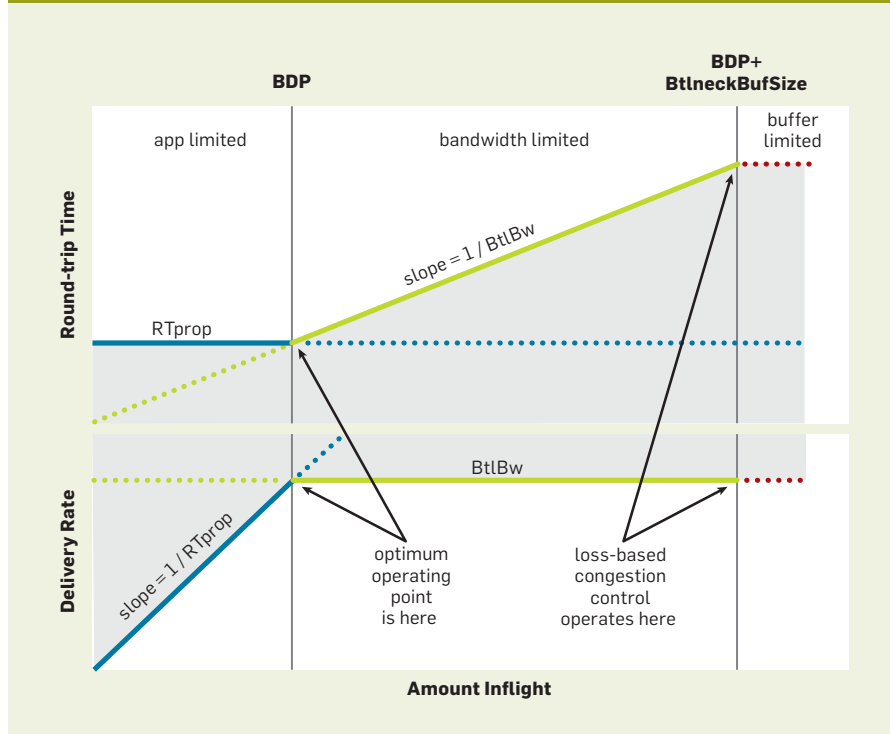


Figure 2. Ack-arrival half of the BBR algorithm.

```
function onAck(packet)
  rtt = now - packet.sendtime
  update_min_filter(RTpropFilter, rtt)
  delivered += packet.size
  delivered_time = now
  deliveryRate = (delivered - packet.delivered) /
    (delivered_time - packet.delivered_time)
  if (deliveryRate > BtlBwFilter.currentMax
      || ! packet.app_limited)
    update_max_filter(BtlBwFilter, deliveryRate)
  if (app_limited_until > 0)
    app_limited_until = app_limited_until - packet.size
```

$$\widehat{RTprop} = RTprop + \min(\eta) = \min(RTT) \forall t \in [T - W_R, T]$$

(That is, a running min over time window W_R (which is typically tens of seconds to minutes).

Unlike RTT, nothing in the TCP spec requires implementations to track bottleneck bandwidth, but a good estimate results from tracking delivery rate. When the ack for some packet arrives back at the sender, it conveys that packet's RTT and announces the delivery of data *inflight* when that packet departed. Average delivery rate between send and ack is the ratio of data delivered to time elapsed: $\text{deliveryRate} = \Delta\text{delivered}/\Delta t$. This rate must be \leq the bottleneck rate (the arrival amount is known exactly so all the uncertainty is in the Δt , which must be \geq the true arrival interval; thus, the ratio must be \leq the true delivery rate, which is, in turn, upper-bounded by the bottleneck capacity). Therefore, a windowed-max of delivery rate is an efficient, unbiased estimator of *BtlBw*:

$$\widehat{BtlBw} = \max(\text{deliveryRate}_i) \forall t \in [T - W_B, T]$$

where the time window W_B is typically six to 10 RTTs.

TCP must record the departure time of each packet to compute RTT. BBR augments that record with the total data delivered so each ack arrival yields both an RTT and a delivery rate measurement that the filters convert to *RTprop* and *BtlBw* estimates.

Note that these values are completely independent: *RTprop* can change (for example, on a route change) but still have the same bottleneck, or *BtlBw* can change (for example, when a wireless link changes rate) without the path changing. (This independence is why both constraints have to be known to match sending behavior to delivery path.) Since *RTprop* is visible only to the left of BDP and *BtlBw* only to the right in Figure 1, they obey an uncertainty principle: whenever one can be measured, the other cannot. Intuitively, this is because the pipe has to be overfilled to find its capacity, which creates a queue that obscures the length of the pipe. For example, an application running a request/response protocol might never send enough data to fill the pipe and

Our group at Google spends hours each day examining TCP packet header captures from all over the world, making sense of behavior anomalies and pathologies. Our usual first step is finding the essential path characteristics, *RTprop* and *BtlBw*.

observe *BtlBw*. A multi-hour bulk data transfer might spend its entire lifetime in the bandwidth-limited region and have only a single sample of *RTprop* from the first packet's RTT. This intrinsic uncertainty means that in addition to estimators to recover the two path parameters, there must be states that track both what can be learned at the current operating point and, as information becomes stale, how to get to an operating point where it can be relearned.

Matching the Packet Flow to the Delivery Path

The core BBR algorithm has two parts:

When an ack is received. Each ack provides new RTT and delivery rate measurements that update the *RTprop* and *BtlBw* estimates, as illustrated in Figure 2.

The `if` checks address the uncertainty issue described in the last paragraph: senders can be application limited, meaning the application runs out of data to fill the network. This is quite common because of request/response traffic. When there is a send opportunity but no data to send, BBR marks the corresponding bandwidth sample(s) as application limited (see `send()` pseudocode to follow). The code here decides which samples to include in the bandwidth model so it reflects network, not application, limits. *BtlBw* is a hard upper bound on the delivery rate so a measured delivery rate larger than the current *BtlBw* estimate must mean the estimate is too low, whether or not the sample was app-limited. Otherwise, application-limited samples are discarded. (Figure 1 shows that in the app-limited region deliveryRate underestimates, *BtlBw*. These checks prevent filling the *BtlBw* filter with underestimates that would cause data to be sent too slowly.)

When data is sent. To match the packet-arrival rate to the bottleneck link's departure rate, BBR paces every data packet. BBR must match the bottleneck *rate*, which means pacing is integral to the design and fundamental to operation—`pacing_rate` is BBR's primary control parameter. A secondary parameter, `cwnd_gain`, bounds *inflight* to a small multiple of the BDP to handle common network and receiver pathologies (as we will discuss). Conceptually, the TCP send routine looks like

Figure 3. Packet-send half of the BBR algorithm.

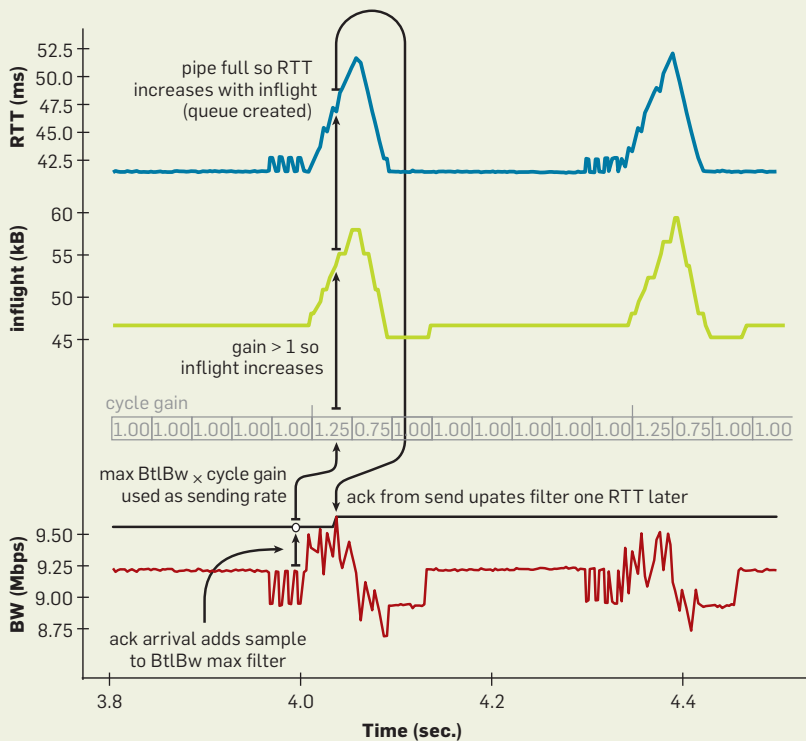
```

function send(packet)
    bdp = BtlBwFilter.currentMax
        * RTpropFilter.currentMin
    if (inflight >= cwnd_gain * bdp)
        // wait for ack or retransmission timeout
        return
    if (now >= nextSendTime)
        packet = nextPacketToSend()
        if (! packet)
            app_limited_until = inflight
            return
        packet.app_limited = (app_limited_until > 0)
        packet.sendtime = now
        packet.delivered = delivered
        packet.delivered_time = delivered_time
        ship(packet)
        nextSendTime = now + packet.size /
            (pacing_gain * BtlBwFilter.currentMax)

    timerCallbackAt(send, nextSendTime)

```

Figure 4. RTT (blue), inflight (green), and delivery rate (red) detail.



the code in Figure 3. (In Linux, sending uses the efficient FQ/pacing queuing discipline,⁴ which gives BBR line-rate single-connection performance on multigigabit links and handles thousands of lower-rate paced connections with negligible CPU overhead.)

Steady-state behavior. The rate and amount BBR sends is solely a function of the estimated *BtlBw* and *RTprop*, so the filters control adaptation in addi-

tion to estimating the bottleneck constraints. This creates the novel control loop shown in Figure 4, which illustrates the RTT (blue), *inflight* (green) and delivery rate (red) detail from 700ms of a 10Mbps, 40ms flow. The thick gray line above the delivery-rate data is the state of the *BtlBw* max filter. The triangular structures result from BBR cycling *pacing_gain* to determine if *BtlBw* has increased. The gain used

for each part of the cycle is shown time-aligned with the data it influenced. The gain is applied an RTT earlier, when the data is sent. This is indicated by the horizontal jog in the event sequence description running up the left side.

BBR minimizes delay by spending most of its time with one BDP in flight, paced at the *BtlBw* estimate. This moves the bottleneck to the sender so it cannot observe *BtlBw* increases. Consequently, BBR periodically spends an *RTprop* interval at a *pacing_gain* > 1, which increases the sending rate and *inflight*. If *BtlBw* hasn't changed, then a queue is created at the bottleneck, increasing RTT, which keeps *deliveryRate* constant. (This queue is removed by sending at a compensating *pacing_gain* < 1 for the next *RTprop*.) If *BtlBw* has increased, *deliveryRate* increases and the new max immediately increases the *BtlBw* filter output, increasing the base pacing rate. Thus, BBR converges to the new bottleneck rate exponentially fast. Figure 5 shows the effect on a 10Mbps, 40ms flow of *BtlBw* abruptly doubling to 20Mbps after 20 seconds of steady operation (top graph) then dropping to 10Mbps after another 20 seconds of steady operation at 20Mbps (bottom graph).

(BBR is a simple instance of a Max-plus control system, a new approach to control based on nonstandard algebra.¹² This approach allows the adaptation rate [controlled by the *max gain*] to be independent of the queue growth [controlled by the *average gain*]. Applied to this problem, it results in a simple, implicit control loop where the adaptation to physical constraint changes is automatically handled by the filters representing those constraints. A conventional control system would require multiple loops connected by a complex state machine to accomplish the same result.)

Single BBR Flow Startup Behavior

Existing implementations handle events such as startup, shutdown, and loss recovery with event-specific algorithms and many lines of code. BBR uses the code detailed earlier for everything, handling events by sequencing through a set of “states” that are defined by a table containing one or more fixed gains and exit criteria. Most of the time is spent in the ProbeBW state described in the section on Steady-state Behavior.

The Startup and Drain states are used at connection start (Figure 6). To handle Internet link bandwidths spanning 12 orders of magnitude, Startup implements a binary search for *BtlBw* by using a gain of $2/\ln 2$ to double the sending rate while delivery rate is increasing. This discovers *BtlBw* in $\log_2 BDP$ RTTs but creates up to $2BDP$ excess queue in the process. Once Startup finds *BtlBw*, BBR transitions to Drain, which uses the inverse of Startup's gain to get rid of the excess queue, then to ProbeBW once the *inflight* drops to a BDP.

Figure 6 shows the first second of a 10Mbps, 40ms BBR flow. The time/sequence plot shows the sender (green) and receiver (blue) progress vs. time. The red line shows a CUBIC sender under identical conditions. Vertical gray lines mark BBR state transitions. The lower figure shows the RTT of the two connections vs. time. Note that the time reference for this data is ack arrival (blue) so, while they appear to be time shifted, events are shown at the point where BBR learns of them and acts.

The lower graph of Figure 6 contrasts BBR and CUBIC. Their initial behavior is similar, but BBR completely drains its startup queue while CUBIC can't. Without a path model to tell it how much of the *inflight* is excess, CUBIC makes *inflight* growth less aggressive, but growth continues until either the bottleneck buffer fills and drops a packet or the receiver's *inflight* limit (TCP's receive window) is reached.

Figure 7 shows RTT behavior during the first eight seconds of the connections shown in Figure 6. CUBIC (red) fills the available buffer, then cycles from 70% to 100% full every few seconds. After startup, BBR (green) runs with essentially no queue.

Behavior of Multiple BBR Flows Sharing a Bottleneck

Figure 8 shows how individual throughputs for several BBR flows sharing a 100Mbps/10ms bottleneck converge to a fair share. The downward facing triangular structures are connection ProbeRTT states whose self-synchronization accelerates final convergence.

ProbeBW gain cycling (Figure 4) causes bigger flows to yield bandwidth to smaller flows, resulting in each learning its fair share. This happens fairly quickly (a few ProbeBW cycles),

though unfairness can persist when late starters overestimate *RTprop* as a result of starting when other flows have (temporarily) created a queue.

To learn the true *RTprop*, a flow moves to the left of BDP using

ProbeRTT state: when the *RTprop* estimate has not been updated (that is, by measuring a lower RTT) for many seconds, BBR enters ProbeRTT, which reduces the *inflight* to four packets for at least one round trip,

Figure 5. Bandwidth change.

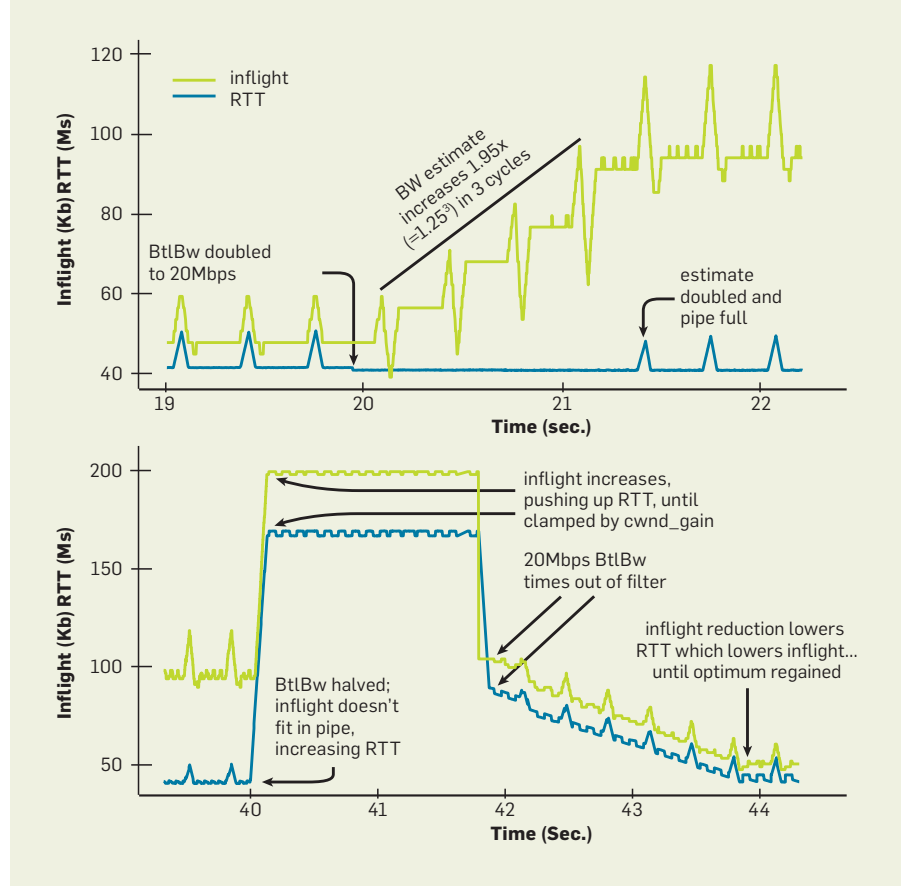
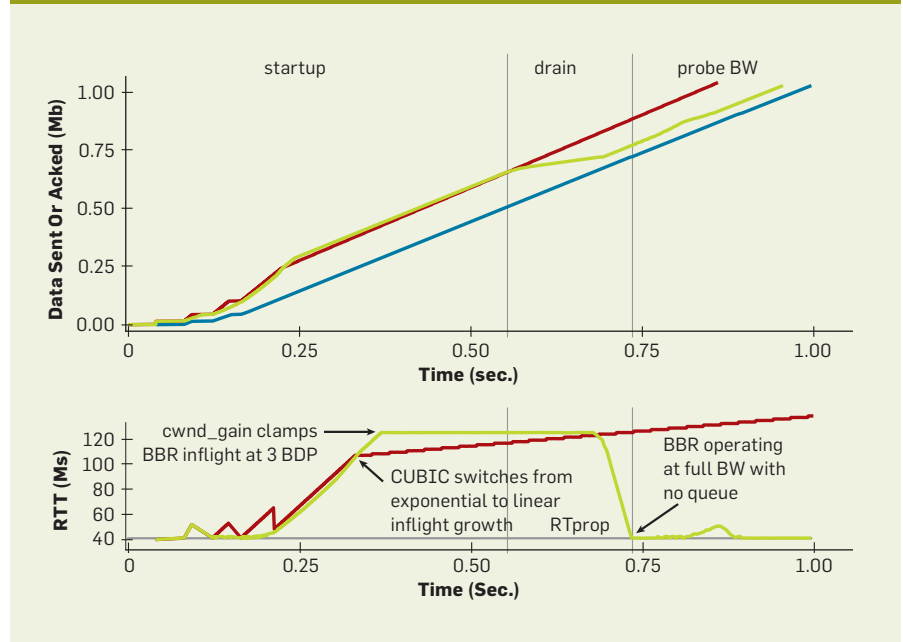


Figure 6. First second of a 10Mbps, 40ms BBR flow



then returns to the previous state. Large flows entering ProberTT drain many packets from the queue, so several flows see a new RT_{prop} (new minimum RTT). This makes their RT_{prop}

estimates expire at the same time, so they enter ProberTT together, which makes the total queue dip larger and causes more flows to see a new RT_{prop} , and so on. This distributed co-

ordination is the key to both fairness and stability.

BBR synchronizes flows around the desirable event of an empty bottleneck queue. By contrast, loss-based congestion control synchronizes around the undesirable events of periodic queue growth and overflow, amplifying delay and packet loss.

Google B4 WAN Deployment Experience

Google's B4 network is a high-speed WAN (wide-area network) built using commodity switches.¹⁵ Losses on these shallow-buffered switches result mostly from coincident arrivals of small traffic bursts. In 2015, Google started switching B4 production traffic from CUBIC to BBR. No issues or regressions were experienced, and since 2016 all B4 TCP traffic uses BBR. Figure 9 shows one reason for switching: BBR's throughput is consistently 2 to 25 times greater than CUBIC's. We had expected even more improvement but discovered that 75% of BBR connections were limited by the kernel's TCP receive buffer, which the network operations team had deliberately set low (8MB) to prevent CUBIC flooding the network with megabytes of excess *inflight* (8MB/200ms intercontinental RTT \Rightarrow 335Mbps max throughput). Manually raising the receive buffer on one U.S.-Europe path caused BBR immediately to reach 2Gbps, while CUBIC remained at 15Mbps—the 133x relative improvement predicted by Mathis et al.¹⁷

Figure 9 shows BBR vs. CUBIC relative throughput improvement; the inset shows throughput CDFs (cumulative distribution functions). Measures are from an active prober service that opens persistent BBR and CUBIC connections to remote datacenters, then transfers 8MB of data every minute. Probers communicate via many B4 paths within and between North America, Europe, and Asia.

The huge improvement is a direct consequence of BBR *not* using loss as a congestion indicator. To achieve full bandwidth, existing loss-based congestion controls require the loss rate to be less than the inverse square of the BDP¹⁷ (for example, < one loss per 30 million packets for a 10Gbps/100ms path). Figure 10 compares measured goodput at various

Figure 7. First eight seconds of 10Mbps, 40ms cubic and BBR flows.

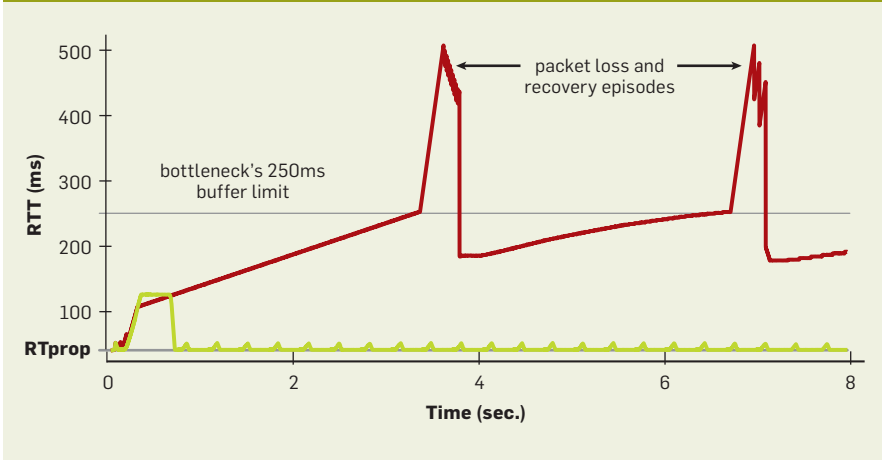


Figure 8. Throughputs of five BBR flows sharing a bottleneck

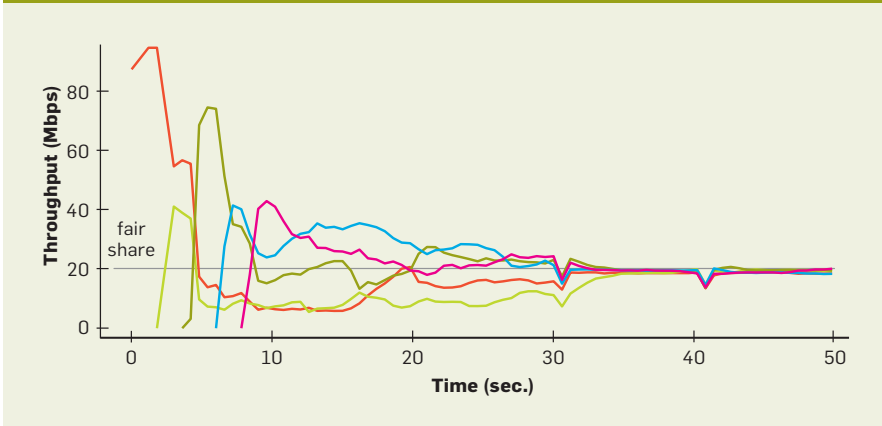
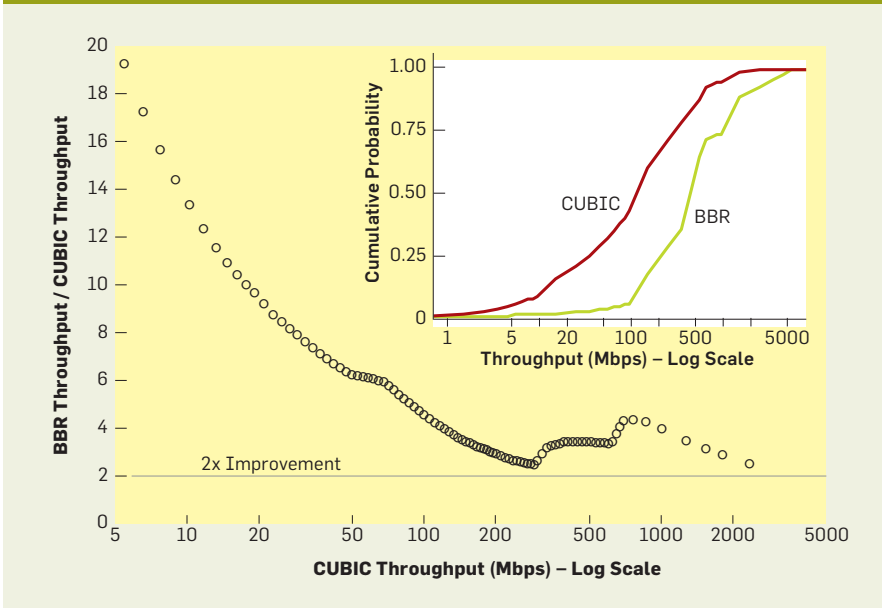


Figure 9. BBR vs. CUBIC relative throughput improvement.



loss rates. CUBIC's loss tolerance is a structural property of the algorithm, while BBR's is a configuration parameter. As BBR's loss rate approaches the ProbeBW peak gain, the probability of measuring a delivery rate of the true $BtBW$ drops sharply, causing the max filter to underestimate.

Figure 10 shows BBR vs. CUBIC goodput for 60-second flows on a 100Mbps/100ms link with 0.001 to 50% random loss. CUBIC's throughput decreases by 10 times at 0.1% loss and totally stalls above 1%. The maximum possible throughput is the link rate times fraction delivered ($= 1 - lossRate$). BBR meets this limit up to a 5% loss and is close up to 15%.

YouTube Edge Deployment Experience

BBR is being deployed on Google.com and YouTube video servers. Google is running small-scale experiments in which a small percentage of users are randomly assigned either BBR or CUBIC. Playbacks using BBR show significant improvement in all of YouTube's quality-of-experience metrics, possibly because BBR's behavior is more consistent and predictable. BBR only slightly improves connection throughput because YouTube already adapts the server's streaming rate to well below $BtBW$ to minimize bufferbloat and rebuffer events. Even so, BBR reduces median RTT by 53% on average globally and by more than 80% in the developing world. Figure 11 shows BBR vs. CUBIC median RTT improvement from more than 200 million YouTube playback connections measured on five continents over a week.

More than half of the world's seven billion mobile Internet subscriptions connect via 8kbps to 114kbps 2.5G systems,⁵ which suffer well-documented problems because of loss-based congestion control's buffer-filling propensities.³ The bottleneck link for these systems is usually between the SGSN (serving GPRS support node)¹⁸ and mobile device. SGSN software runs on a standard PC platform with ample memory, so there are frequently megabytes of buffer between the Internet and mobile device. Figure 12 compares (emulated) SGSN Internet-to-mobile delay for BBR and CUBIC.

Figure 10. BBR vs. CUBIC goodput under loss.

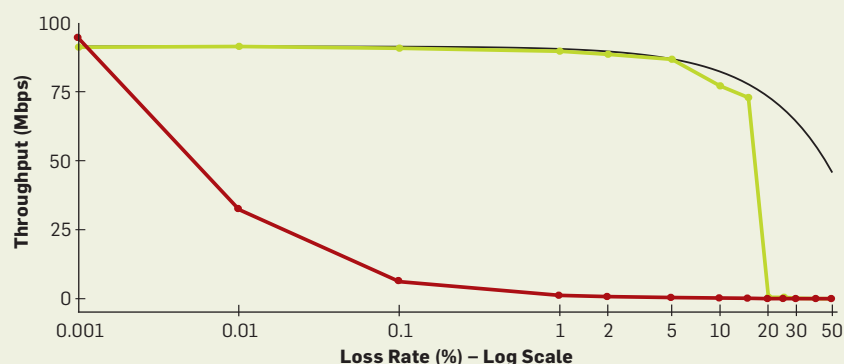


Figure 11. BBR vs. CUBIC median RTT improvement.

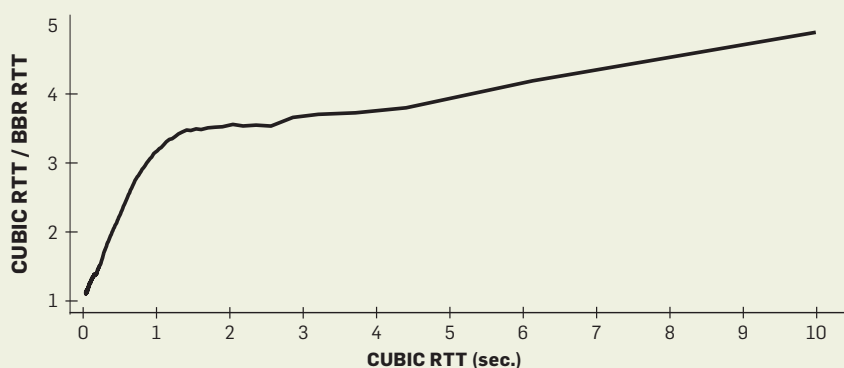
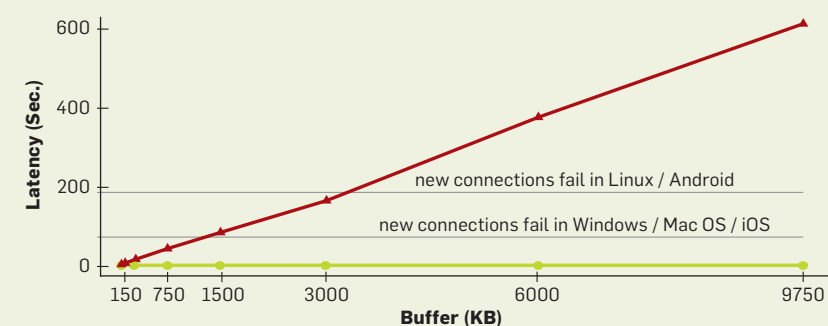


Figure 12. Steady-state median RTT variation with link buffer size



The horizontal lines mark one of the more serious consequences: TCP adapts to long RTT delay except on the connection initiation SYN packet, which has an OS-dependent fixed timeout. When the mobile device is receiving bulk data (for example, from automatic app updates) via a large-buffered SGSN, the device cannot connect to anything on the Internet until the queue empties (the SYN ACK ac-

cept packet is delayed for longer than the fixed SYN timeout).

Figure 12 shows steady-state median RTT variation with link buffer size on a 128Kbps/40ms link with eight BBR (green) or CUBIC (red) flows. BBR keeps the queue near its minimum, independent of both bottleneck buffer size and number of active flows. CUBIC flows always fill the buffer, so the delay grows linearly with buffer size.

Mobile Cellular Adaptive Bandwidth

Cellular systems adapt per-subscriber bandwidth based partly on a demand estimate that uses the queue of packets destined for the subscriber. Early versions of BBR were tuned to create very small queues, resulting in connections getting stuck at low rates. Raising the peak ProbeBW pacing_gain to create bigger queues resulted in fewer stuck connections, indicating it is possible to be too nice to some networks. With the current $1.25 \times BtlBw$ peak gain, no degradation is apparent compared with CUBIC on any network.

Delayed and stretched acks. Cellular, Wi-Fi, and cable broadband networks often delay and aggregate ACKs.¹ When *inflight* is limited to one BDP, this results in throughput-reducing stalls. Raising ProbeBW's cwnd_gain to two allowed BBR to continue sending smoothly at the estimated delivery rate, even when ACKs are delayed by up to one RTT. This largely avoids stalls.

Token-bucket policers. BBR's initial YouTube deployment revealed that most of the world's ISPs mangle traffic with token-bucket policers.⁷ The bucket is typically full at connection startup so BBR learns the underlying network's *BtlBw*, but once the bucket empties, all packets sent faster than the (much lower than *BtlBw*) bucket fill rate are dropped. BBR eventually learns this new delivery rate, but the ProbeBW gain cycle results in continuous moderate losses. To minimize the upstream bandwidth waste and application latency increase from these losses, we added policer detection and an explicit policer model to BBR. We are also actively researching better ways to mitigate the policer damage.

Competition with loss-based congestion control. BBR converges toward a fair share of the bottleneck bandwidth whether competing with other BBR flows or with loss-based congestion control. Even as loss-based congestion control fills the available buffer, ProbeBW still robustly moves the *BtlBw* estimate toward the flow's fair share, and ProbeRTT finds an *RTTprop* estimate just high enough for tit-for-tat con-


vergence to a fair share. Unmanaged router buffers exceeding several BDPs, however, cause long-lived loss-based competitors to bloat the queue and grab more than their fair share. Mitigating this is another area of active research.

Conclusion

Rethinking congestion control pays big dividends. Rather than using events such as loss or buffer occupancy, which are only weakly correlated with congestion, BBR starts from Kleinrock's formal model of congestion and its associated optimal operating point. A pesky "impossibility" result that the crucial parameters of delay and bandwidth cannot be determined simultaneously is sidestepped by observing they can be estimated sequentially. Recent advances in control and estimation theory are then used to create a simple distributed control loop that verges on the optimum, fully utilizing the network while maintaining a small queue. Google's BBR implementation is available in the open source Linux kernel TCP.

BBR is deployed on Google's B4 backbone, improving throughput by orders of magnitude compared with CUBIC. It is also being deployed on Google and YouTube Web servers, substantially reducing latency on all five continents tested to date, most dramatically in developing regions. BBR runs purely on the sender and does not require changes to the protocol, receiver, or network, making it incrementally deployable. It depends only on RTT and packet-delivery acknowledgment, so can be implemented for most Internet transport protocols.

Acknowledgments

Thanks to Len Kleinrock for pointing out the right way to do congestion control and Larry Brakmo for pioneering work on Vegas² and New Vegas congestion control that presaged many elements of BBR, and for advice and guidance during BBR's early development. We also thank Eric Dumazet, Nandita Dukkupati, Jana Iyengar, Ian Swett, M. Fitz Nowlan, David Wetherall, Leonidas Kontothanassis, Amin Vahdat, and the Google BwE and YouTube infrastructure teams. 

Related articles on queue.acm.org

Sender-side Buffers and the Case for Multimedia Adaptation

Aiman Erbad and Charles "Buck" Krasic
<http://queue.acm.org/detail.cfm?id=2381998>

You Don't Know Jack about Network Performance

Kevin Fall and Steve McCanne
<http://queue.acm.org/detail.cfm?id=1066069>

A Guided Tour through Data-center Networking

Dennis Abts and Bob Felderman
<http://queue.acm.org/detail.cfm?id=2208919>

References

1. Abrahamsson, M. TCP ACK suppression. IETF AQM mailing list, 2015; <https://www.ietf.org/mail-archives/web/aqm/current/msg01480.html>.
2. Brakmo, L.S., Peterson, L.L. TCP Vegas: End-to-end congestion avoidance on a global Internet. *IEEE J. Selected Areas in Communications* 13, 8 (1995), 1465–1480.
3. Chakravorty, R., Cartwright, J., Pratt, I. Practical experience with TCP over GPRS. *IEEE GLOBECOM*, 2002.
4. Corbet, J. TSO sizing and the FQ scheduler. LWN.net, 2013; <https://lwn.net/Articles/564978/>.
5. Ericsson. Ericsson Mobility Report (June), 2015; <https://www.ericsson.com/res/docs/2015/ericsson-mobility-report-june-2015.pdf>.
6. ESnet. Application tuning to optimize international astronomy workflow from NERSC to LFI-DPC at INAF-OATs; <http://fasterdata.es.net/data-transfer-tools/case-studies/nersc-astronomy/>.
7. Flach, T. et al. An Internet-wide analysis of traffic policing. *SIGCOMM*, 2016, 468–482.
8. Gail, R., Kleinrock, L. An invariant property of computer network power. In *Proceedings of the International Conference on Communications* 63, 1 (1981), 1–63.15.
9. Gettys, J., Nichols, K. Bufferbloat: Dark buffers in the Internet. *acmqueue* 9, 11 (2011); <http://queue.acm.org/detail.cfm?id=2071893>.
10. Ha, S., Rhee, I. 2011. Taming the elephants: new TCP slow start. *Computer Networks* 55, 9 (2011), 2092–2110.
11. Ha, S., Rhee, I., Xu, L. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review* 42, 5 (2008), 64–74.
12. Heidergott, B., Olsder, G. J., Van Der Woude, J. *Max Plus at Work: Modeling and Analysis of Synchronized Systems: a Course on Max-Plus Algebra and its Applications*. Princeton University Press, 2014.
13. Jacobson, V. Congestion avoidance and control. *ACM SIGCOMM Computer Communication Review* 18, 4 (1988): 314–329.
14. Jaffe, J. Flow control power is nondecentralizable. *IEEE Transactions on Communications* 29, 9 (1981), 1301–1306.
15. Jain, S. et al. B4: experience with a globally-deployed software defined WAN. *ACM SIGCOMM Computer Communication Review* 43, 4 (2013), 3–14.
16. Kleinrock, L. 1979. Power and deterministic rules of thumb for probabilistic problems in computer communications. In *Proceedings of the International Conference on Communications* (1979), 43.1.1–43.1.10.
17. Mathis, M., Semke, J., Mahdavi, J., Ott, T. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review* 27, 3 (1997), 67–82.
18. Wikipedia. GPRS core network serving GPRS support node; https://en.wikipedia.org/wiki/GPRS_core_network#Serving_GPRS_support_node_28SGSN.29.

Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson are members of Google's make-top-fast project, whose goal is to evolve Internet transport via fundamental research and open source software. Project contributions include TFO (TCP Fast Open), TLP (Tail Loss Probe), RACK loss recovery, fq/pacing, and a large fraction of the git commits to the Linux kernel TCP code for the past five years. They can be contacted at <https://googlegroups.com/d/forum/bbr-dev>.

Copyright held by owners/authors.

ACM Thanks Chapters for Participating in Hour of Code

Week-long event inspires next generation of computer scientists

The Hour of Code is a global movement designed to generate excitement in young people about programming and technology. Games, tutorials, and other events were organized during Computer Science Education Week around the world. ACM would like to thank the following chapters who participated this year:

- Ain Shams University ACM Student Chapter
- Anna University School of CS&Eng ACM Student Chapter
- Bethel University ACM Student Chapter
- Bharati Vidyapeeth's College of Engineering ACM Student Chapter
- Bilkent University ACM Student Chapter
- Bilkent University ACM-W Student Chapter
- Boise State University ACM-W Student Chapter
- Chitkara ACM Student Chapter
- Clovis ACM Student Chapter
- County College of Morris ACM Student Chapter
- Holy Family University ACM Student Chapter
- IBA ACM Student Chapter
- IIT (ISM) Dhanbad ACM Student Chapter
- Koc University ACM Student Chapter
- Letterkenny Institute ACM-W Student Chapter
- LYIT ACM Student Chapter
- Manipal University Jaipur ACM Student Chapter
- McMaster University ACM Student Chapter
- MIET ACM Student Chapter
- Montana State University ACM-W Student Chapter
- Norfolk State University
- NUST ACM Student Chapter
- Oregon State University ACM-W Student Chapter
- Pomona College ACM-W Student Chapter
- PUCIT ACM Student Chapter
- Puerto Rico Rio Piedras ACM Student Chapter
- Sheridan CSS ACM Student Chapter
- Snow College ACM-W Student Chapter
- Sri Aurobindo Institute of Technology ACM Student Chapter
- Sukkur IBA ACM Student Chapter
- Touro College
- UCSP ACM Student Chapter
- UMass Lowell ACM Student Chapter
- University of Louisiana/Monroe
- University of Nebraska-Lincoln ACM-W Student Chapter
- University of Texas at Austin
- University of Toronto iSchool ACM Student Chapter
- UPES ACM Student Chapter
- UPES ACM-W Student Chapter



Association for
Computing Machinery

DOI:10.1145/2979673

Government-sanctioned and market-based anti-piracy measures can both mitigate economic harm from piracy.

BY BRETT DANAHER, MICHAEL D. SMITH, AND RAHUL TELANG

Copyright Enforcement in the Digital Age: Empirical Evidence and Policy Implications

THE CORRELATION BETWEEN the rise of Internet-based piracy^a and drops in revenue from media sales has made online copyright infringement a hotly debated topic in the media and technology industries. In the 10 years following Napster's introduction in 1999, global recorded music sales decreased 50%, despite having previously been on an upward trend.¹⁶ Likewise, while DVD/VHS sales increased from 2000 to 2003, sales fell 27% in the four years after the widespread adoption of the BitTorrent protocol 2004.

a Throughout this article, when we use the term "piracy" in the context of digital media consumption, we follow the *Oxford English Dictionary* definition of "the unauthorized use or reproduction of another's work."

Although many industry observers approach this topic with strong, and fundamentally philosophical, viewpoints, a community of academic researchers has aimed to bring objective and robust empirical analysis to the measurement and analysis of illegal online filesharing. The first phase of this research focused primarily on the impact of digital piracy on legal sales. We are aware of 26 peer-reviewed journal articles studying the economic harm caused by piracy, with 23 of them finding piracy causes significant harm to legal sales. In short, "the dust has settled in that literature," as Joel Waldfoegel of the University of Minnesota observed at a 2015 meeting of the World Intellectual Property Organization, "... and most people believe that, indeed, unpaid consumption reduces the ability of sellers to generate revenues."^{b,c}

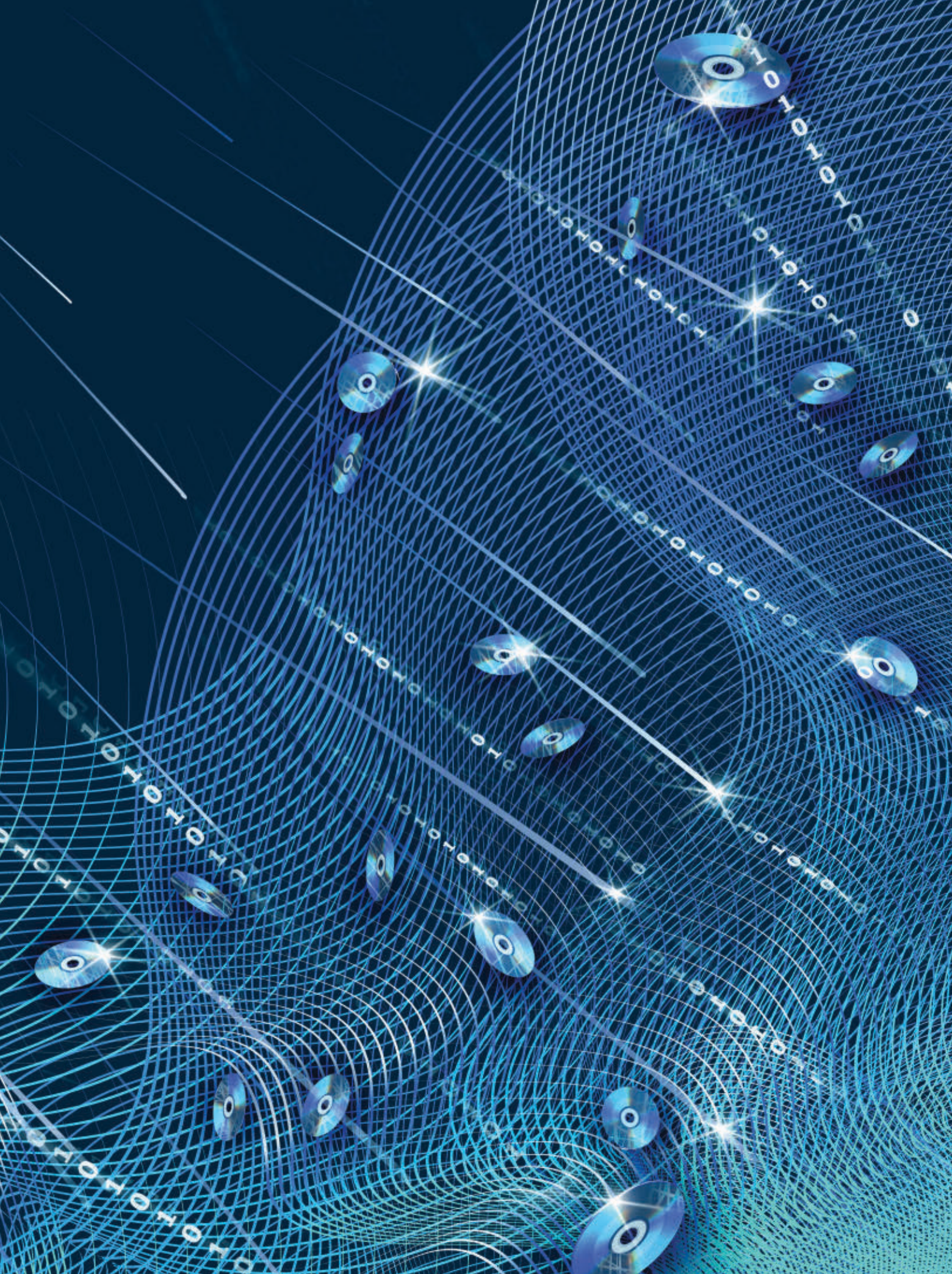
With the harm from piracy well established in the literature, our goal

b See <https://www.youtube.com/watch?v=D0gwM2WRjGE#t=9m20s>. See also Robert Hammond's observation in his 2014 peer-reviewed paper in the *Southern Economic Journal* that it is a "well-documented fact that file sharing is harmful to the music industry." Literature reviews by Danaher et al.,⁷ Liebowitz,¹⁶ and Oberholzer-Gee and Strumpf¹⁷ all draw similar conclusions.

c There is also a small but inconclusive literature on whether and how revenue lost due to piracy affect the supply of creative works, as in Telang and Waldfoegel²³ and Waldfoegel.²⁴

» key insights

- **The empirical evidence shows the best way to reduce the economic harm caused by digital piracy is through combined anti-piracy efforts from rightsholders, technology firms, and governments.**
- **Rightsholders can reduce piracy and increase legal sales by making legal content more easily accessible in digital channels and by synchronizing international release dates; technology firms can respond by making pirated content more difficult to find on digital platforms.**
- **Governments can respond to piracy by introducing well-known and well-enforced legal penalties against individual pirates, by taking legal action to shut down or otherwise block Internet access to prominent pirate sites.**




here is to synthesize the findings in the more recent literature that analyzes the effectiveness of various approaches for reducing the economic harm from piracy. Some reviews in the literature explore the underlying determinants of a consumer's decision to acquire content legally or illegally; see, for example, Watson et al.²⁵ Other papers explore the question through a laboratory experiment, as in, for example Fleming et al.¹² Unlike this work, we focus specifically on evaluating the effectiveness of a number of strategies rightsholders can use to respond to piracy, including making legal content more available or convenient, prioritizing links to legal sites in search results, and removing title-specific copies of media files from the Internet. We then analyze the results from various studies of specific government anti-piracy interventions, synthesizing these results to provide insights into the determinants of the effectiveness of such interventions.

We show that firm strategies and government interventions can both have meaningful effects on consumer behavior, reducing piracy and increasing legal sales. However, we also find that no single action is a panacea for the problem posed by piracy. We conclude that the most effective response to piracy involves combined efforts from both rightsholders and governments.


Business Strategies

Rightsholders have adopted a number of strategies in an attempt to persuade pirates to consume their goods through legal channels. Such strategies often come at a cost to the firm, and measuring the effectiveness of these strategies is thus of great importance.

Increasing the availability and convenience of legal distribution channels appears to be a significant factor in reducing piracy. For example, Danaher et al.¹⁰ showed that NBC's decision to remove its television content from the iTunes video store on December 1, 2007 caused piracy of that content to increase by 11% relative to a control group of content from other television networks.¹⁰ Figure 1 displays piracy of NBC vs. non-NBC content before and after NBC removed its content from iTunes. Furthermore, when NBC later restored its content to the iTunes store, piracy diminished. Danaher et al.¹⁰



We are aware of 26 peer-reviewed journal articles studying the economic harm caused by piracy, with 23 of them finding piracy causes significant harm to legal sales.



also found that neither of these events caused any statistically significant change in physical DVD sales of that content, a notable observation given that the desire not to cannibalize physical sales is a common reason firms give for delaying adoption of digital distribution channels.

In a related paper, Danaher et al.⁵ showed that ABC's decision to add some of its television programs to Hulu caused a 20% decrease in piracy of that content, implying that offering content in a convenient way (on a digital-subscription or ad-supported service) can convert a significant number of pirates to legal consumption.

Zhang²⁶ examined the removal of digital rights management (DRM) protection from the catalog of EMI Music—one of the four major music labels at the time—and found it was associated with an increase in EMI's digital music sales relative to changes in the other labels' sales, and that the increase in sales was larger for less-popular content than for more-popular content. Given the consistent finding in the literature that piracy diminishes sales of copyrighted works, a reasonable interpretation of this result is that the increased appeal and utility of DRM-free content convinced some pirates to purchase legally.

Other studies have shown that reducing the time between the U.S. release of a film and its international release in theaters (Danaher and Waldfogel⁹) or on DVD (Smith and Telang²²) can decrease piracy and increase sales.

In addition to making legal content more available, rightsholders can also make pirated content less appealing or less available. Christin et al.³ showed that “poisoning” filesharing networks with replaced decoy files can manipulate consumer perceptions of content availability on a piracy network, but recent piracy technologies have included safeguards against such actions, and this strategy is no longer commonly used. Reimers²⁰ documented that when publishers employed a third-party organization to selectively increase copyright enforcement on a specific set of book titles by having Google de-list the sites offering the copyright-infringing files and by sending takedown notices to those sites, this action caused ebook sales of those

titles to increase by 11%.^d However, such private enforcement of public copyright policy is controversial and the subject of much legal debate; see, for example, The Takedown Project (<http://takedownproject.org/projects>) and Kuczerawy.¹⁴

Beyond strategies individual firms can pursue, there may be opportunities to protect content through industry cooperation. One example on the demand side is the Copyright Alert System in the U.S. in which many ISPs have voluntarily agreed to a graduated response system of warnings and penalties when they detect copyright infringement by their users. We are aware of no academic evidence as to whether this system has had any impact. On the supply side, however, we have studied search behavior in relation to piracy and found that demoting search results that link to piracy websites can shift user behavior toward legal consumption, implying search engines may be useful partners in the effort to reduce piracy's impact.²¹

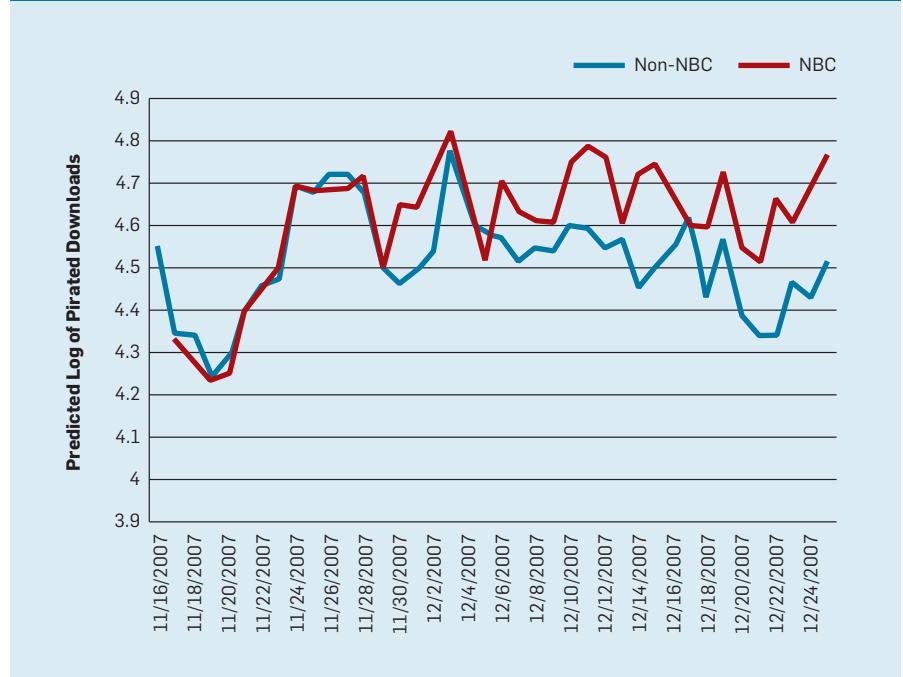
In summary, there is strong evidence that rightsholders can reduce piracy and increase legal consumption by offering their content in more convenient channels and by reducing de-

lays in availability between countries and among distribution channels. There is also evidence that cooperation from firms outside the entertainment industry can help protect copyrighted content. However, each of these strategies is costly to the firms involved, and none of them can fully mitigate the impact of piracy on sales. It is thus worth considering whether government enforcement might also serve to mitigate the impact of piracy on legal consumption.

Enforcement

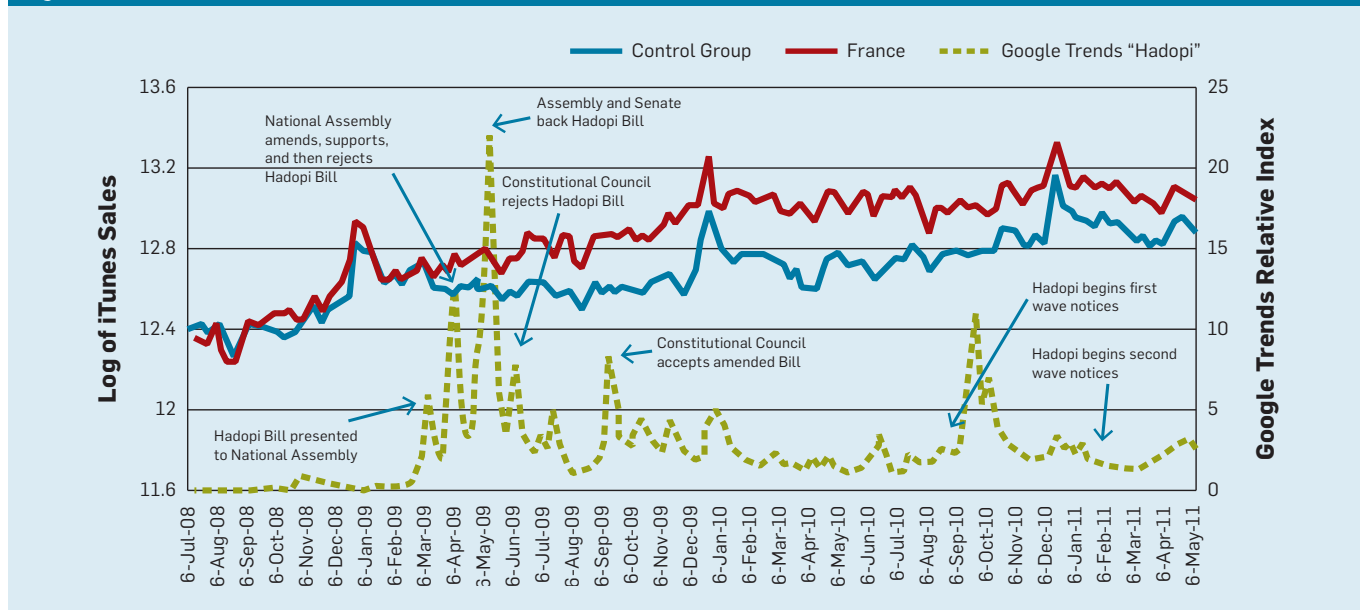
The digitization of media goods effectively weakened copyright laws across the globe by making it easy for ordinary consumers to illegally share media files from computer to computer. Many industry observers have called for reform of existing copyright policy to address issues particular to digitization, and governments have tried a variety of policies to mitigate the impact piracy has had on sales. By analyzing the effectiveness of these diverse efforts at copyright

Figure 1. NBC vs. non-NBC piracy before and after NBC removed its content from iTunes, December 1, 2007; source: Danaher et al.¹⁰



^d Although this is an example of private copyright enforcement, it is worth noting that such private enforcement is possible only in a policy environment where copyright infringement is illegal.

Figure 2. iTunes music sales before and after HADOPI was introduced.



enforcement we can identify and understand the principles behind which policies are most effective. We start by dividing anti-piracy policies as either having a demand-side or supply-side focus.

Demand-side anti-piracy. Demand-side anti-piracy policies focus on enforcement by targeting individuals engaged in illegal downloading of copyrighted works, either with penalties for said illegal behavior or with positive incentives for legal consumption.

HADOPI. The HADOPI law, or Creation and Internet law, a graduated-response anti-piracy law passed in by the French government in 2009, was one of the first demand-side policies enacted in response to piracy. The law empowered the French HADOPI authority (in English, the High Authority for Transmission of Creative Works and Copyright Protection on the Internet) to send warnings to identified copyright infringers and, after repeated infringement, refer the case to courts to impose penalties. HADOPI also provided for a number of positive educational efforts aimed at informing consumers of and steering them toward legal options.

HADOPI went through a great deal of political debate in France from March 2009 until being passed into law in September 2009. In Danaher et al.,⁸ we used the event of the law's enactment to measure the effectiveness of the law in migrating music pirates toward legal digital downloads on the iTunes music store. Specifically, we identified a group of countries that had digital

music sales trends similar to France's before HADOPI^e and compared their sales trends before and after HADOPI to the French sales trend.

Figure 2 plots iTunes music sales trends for music sales in France (red) and for the “control group” countries (blue), demonstrating that from July 2008 until March 2009 France's trend was statistically indistinguishable from the control group. The green dashed line indicates French Google searches for the term “HADOPI” and is our measure of French awareness of the law. From March to June 2009, while the law was still under political debate, public awareness of the law rose and spiked. During this period, French music sales began to rise above sales in the control group, and the gap widened as awareness grew. Notably, the increase in French sales began before the law was actually in effect and being enforced but at the same time as the public became aware of the law and the potential penalties it involved. The study showed that HADOPI caused digital music sales to increase approximately 25% relative to the control group, with larger increases for the most-heavily pirated genres, and smaller increases for the least-pirated genres.^f Danaher et al.⁸ also found that the effect of the law appears to have

e Belgium, Germany, Italy, Spain, and the U.K., the five largest iTunes music markets in Europe at the time, other than France.

f As discussed here, this result is robust to de-trending the data by country.

been maintained for more than two years after the public's initial awareness, although it may have diminished slightly during the last few months of the study. The HADOPI agency sent out many infringement warnings from 2010 to 2012 that may have contributed to continued awareness of the law and its continued effectiveness.^g

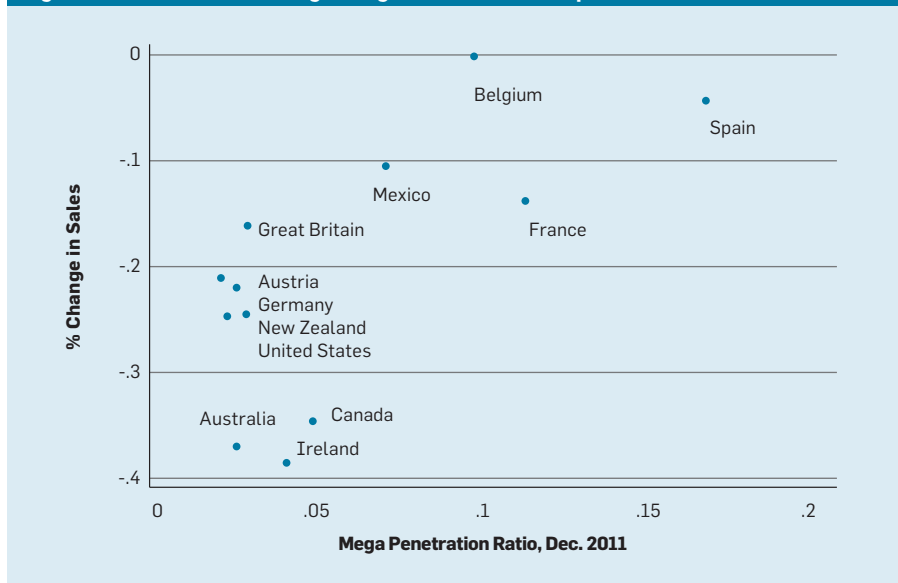
IPRED. In April 2009, Sweden implemented a copyright reform policy based on the European Union's Intellectual Property Rights Enforcement Directive (IPRED) that made it significantly easier for rightsholders to detect and identify filesharers, increasing the risk of punishment for online piracy. Adermon and Liang¹ compared piracy levels and total music sales in Sweden before and after the law to those in two other Scandinavian countries—Norway and Finland. They found the law directly led to a 16% decrease in Internet traffic during the first six months, which they attributed to a 32% decrease in piracy. They also found that total music sales increased 36% during this time relative to the control group, with a larger increase for digital sales and a smaller increase for physical sales. They thus found that awareness of IPRED effectively migrated many music pirates to legal channels.

However, Adermon and Liang¹ also noted that law was enforced very weakly, with only a few cases making it to the courts. After the first six months, piracy levels and music sales both returned to near their original levels, and Adermon and Liang suggest that the transitory effectiveness of the law might be attributed to waning public belief in its level of active enforcement.

Demand-side policies. France and Sweden are not the only countries with demand-side anti-piracy policies. Various forms of demand-side anti-piracy laws exist in Ireland, New Zealand, the Republic of Korea, and the U.K., among others, but we are not aware of peer-reviewed empirical studies on their effectiveness. However, the two aforementioned studies, Adermon and Liang¹ and Danaher et al.,⁸ do share several themes. First, the measured increase in

g This effect is robust to controlling for French iOS—iPhone, iPod, and iPad—sales; see Danaher et al.⁸ and the blog InfoJustice.org (<http://infojustice.org/archives/8891>) for a detailed discussion.

Figure 3. Post-shutdown change in digital movie sales vs. pre-shutdown MPR.



legal consumption was relatively similar (25% vs. 36%) despite the laws being passed at different times and in different countries.^h Second, HADOPI affected behavior when the public became aware of its existence and before it actually took effect, while IPRED's effect diminished after the public observed a lack of enforcement. With demand-side anti-piracy enforcement, awareness of the policy and an expectation of its enforcement appear to be necessary conditions for effectiveness. When these conditions are met, it appears that demand-side policies significantly reduce piracy and increase legal consumption and revenues. However, we cannot say whether the effects of such policies are sustained in the long run. In addition to the IPRED effect disappearing due to weak enforcement, the HADOPI effect appeared to diminish somewhat after the first 18 months, although this decrease was not statistically significant.

Supply-Side Anti-Piracy

Supply-side anti-piracy policies involve targeting sites or protocols that supply access to pirated content. Sources of copyright-infringing files can be either shut down entirely or blocked in a given region in cases where ISPs are ordered to block access to piracy websites. The effectiveness of such policies appears to depend on how inconvenient they make further piracy.

Megaupload shutdown. Cyberlockers, one of the primary means of sharing copyright-infringing files on the Internet, are simply cloud storage space where people can house their data on remote servers. However, some cyberlockers have policies that heavily promote illegal filesharing (such as a lack of passwords to protect account access or cash payments to incentivize individuals who upload popular files). In 2011, the most popular piracy cyberlocker worldwide was <http://www.megaupload.com>, which provided more than 25 petabytes of user uploaded—and largely copyright-infringing—content and accounted for 4% of all Internet traffic.¹⁸ In January 2012, the U.S. Department

^h Both studies involved counterfactual exercises, using a control group to simulate what would have happened in the treated countries if not for the treatments; such difference-in-difference exercises are naturally reliant on the quality of the control group.



Rightsholders have many options for mitigating the impact of piracy on sales, although these strategies often come at a cost to the firms in terms of undermining the effectiveness of their existing marketing strategies.



of Justice shut down Megaupload.com, seizing all of its servers and computer assets, effectively removing its content from the Internet. Many other piracy sites, including its sibling streaming site Megavideo.com (<http://www.megavideo.com>) had linked to the content on this site. Lauinger et al.¹⁵ showed the Megaupload shutdown did not change the set of content available to pirate on the Internet, because the content that existed on Megaupload was also available on other piracy sites, and new piracy sites emerged in its wake.¹⁴

But even though the Megaupload shutdown may not have altered the range of pirated content available, it may still have affected consumer behavior if the remaining content was lower quality or less trustworthy (in terms of being safe and virus free) or if it increased consumer search costs by causing consumers to need to identify and learn to use other sites. In Danaher and Smith,⁶ we asked whether the shutdown of Megaupload increased digital movie sales and rentals. We noted that the Megaupload penetration rate (MPR), or the percent of all Internet users who visited Megaupload in the months before the shutdown, varied significantly across countries. This means the shutdown delivered a larger “shock” to high-MPR countries than to low-MPR countries. To determine the causal effect of the shutdown, we had to determine whether digital movie sales increased more in high-MPR countries than in low-MPR countries after the site was shut down.

Figure 3 demonstrates that countries with high MPR (such as Belgium and Spain) had greater increases in digital movie sales after the shutdown than did countries with lower MPR (such as Australia and Canada).ⁱ We observed a similar pattern for rentals and also that in the months before the shutdown a similar pattern did not exist; that is, there was no relationship between sales trends and MPR until

ⁱ Note the shutdown of Megaupload took place in January 2012, so sales were at the time naturally declining from their holiday peaks. But in countries where the Megaupload shutdown created a larger shock, sales did not decrease by as much as they did in other countries, and in the countries where Megaupload was used the most, sales actually increased after the holidays despite the fact that in other years these countries experienced declines over the same period.

Causal increase in paid legal streaming resulting from U.K. site blocks, November 2013.

Consumer Segment	Pre-block Visits/User to Blocked Sites	Causal Increase in Visits to Legal Sites
0	0	0.0%
1	1.0	2.2%
2	2.0	4.4%
3	3.0	6.5%
4	4.0	8.7%
5	5.4	11.7%
6	8.2	17.5%
7	13.2	26.8%
8	23.8	42.4%
9	66.4	14.8%

the shutdown occurred. We concluded that the shutdown of Megaupload thus caused global revenues from digital movie sales and rentals to increase by 6.5% to 8.5%. However, our data extended only 18 weeks after the shutdown, so it was unclear how long the effect lasted after this 18-week period.

U.K. site blocking. Unlike the Megaupload shutdown, which shut down the entire site worldwide, site blocking involves requiring ISPs in a given country to block access to infringing sites. As a result, the content on these sites is still available on the Internet; it just cannot be accessed through an ISP’s service without some additional measures (such as accessing proxy sites dedicated to providing unblocked access or by using virtual private networks that make it appear a user is accessing a site from a different country).

In May 2012, the U.K. courts ordered ISPs to block access to The Pirate Bay, a major indexing site for BitTorrent tracker files. To study the effectiveness of this program, we obtained data from an Internet consumer panel tracking company on monthly visits to piracy sites and visits to paid legal video streaming sites.⁴ We divided consumers into 10 different segments, with the first segment being non-users of The Pirate Bay, the second being the lightest users of The Pirate Bay, proceeding all the way up to the 10th segment, the heaviest users of The Pirate Bay. Presumably, the block had no effect on non-users of the site (making them a control group) and had an incrementally stronger effect on groups with heavier users of the blocked site than groups

with lighter users. We asked how these groups changed their downloading behavior relative to the control group after The Pirate Bay was blocked.^j

We found when the Pirate Bay was the only site blocked, former users generally increased use of other piracy sites and VPNs, thus causing only a small decrease in total piracy. This finding is consistent with Poort et al.¹⁹ who found only small decreases in total piracy when Dutch ISPs blocked access to The Pirate Bay. Not surprisingly, we found no causal increase in use of paid legal streaming sites from this block; that is, in spite of its popularity, when U.K. ISPs blocked only The Pirate Bay there was no statistical increase in legal consumption. It appears that users simply found other ways to access the same pirated content on other popular sites. A study by Aguiar et al.² using similar methods found similar results in that the shutdown of a single video-linking site Kino.to (now defunct) in Germany did not cause a meaningful increase in legal consumption. Notably, this shutdown was more like a website block than an actual site shutdown, because Kino.to linked to content only on other sites and did not host content; shutting down Kino.to simply removed, or “blocked,” a means

^j Results from this 2015 study were therefore generated by 20 data points, an observation for each of 10 segments both before and after the blocks. However, each of these data points actually represents a sample mean based on the behavior of hundreds or thousands of individuals in the segment, and the resulting tests thus had much greater statistical precision than if the results were based on only the before-and-after behaviors of 10 individuals.

of accessing the content without actually removing the content itself from the Internet. This finding is thus similar to our findings regarding the blocking of The Pirate Bay.

The fact that shutting down a single site apparently has no significant impact on the consumption of piracy could mean either that website blocking is inherently ineffective or that blocking only a small number of sites is insufficient to cause consumers to change their behavior. To test these hypotheses, we analyzed whether a larger number of blocks would have a different impact on consumer behavior than a single block would have. We did this using an event in November 2013 whereby courts in the U.K. ordered the near-simultaneous blocking of 28 piracy sites, with 19 hosting video content. We applied the same methodology to study the effect of these 19 blocks as we used to study the blocking of The Pirate Bay and found a different result: When 19 sites were blocked by ISPs in the U.K. there was a significant reduction in overall piracy, and segments with greater usage of the blocked sites before their blocks exhibited a greater increase in use of legal sites after the blocks occurred,^k a correlation that did not exist prior to the blocks (see the accompanying table).

These results suggest that blocking 19 sites in the U.K. caused average treated consumers to increase their visits to legal sites by 12%. The large decrease in total piracy and the 12% increase in legal site visits demonstrate that, in spite of the fact that some users chose at the time to use VPN services to circumvent the blocks,^l the blocking of 19 major sites did indeed cause a significant shift from illegal to legal consumption.

Supply-side policies. These results suggest that the success of supply-side anti-piracy interventions rests on how inconvenient they make piracy, a view also espoused in a 2015 theory paper by Dey et al.¹¹ Opponents of supply-side

^k The one exception to this is segment 9, as in the Table 6, the heaviest users of the blocked sites, and is discussed further in Danaher et al.⁴

^l The 19-site block resulted in a large-percent-age increase in visits to VPN sites. However, on a unit basis the increase in use of legal sites was much greater than the increase in VPN usage. As noted in Danaher et al.,⁴ this difference is because VPN usage was much less than legal usage prior to the block.

piracy interventions argue that targeting piracy websites cannot be effective, as pirates will always find other sites on which to illegally acquire media, but the empirical research suggests a more nuanced view. Multiple studies document that shutting down or blocking websites may not reduce the range of content available to pirate and that weaker supply-side interventions (such as blocking only one site or shutting down a linking site) cause only small reductions in piracy and do not increase legal activity.

However, several studies show that significant supply-side actions (such as the simultaneous blocking of many sites or the complete shutdown of a large, international piracy site that hosted content) can both reduce total piracy levels and increase consumption through legal channels. This happens even if the blocked content is available on other less-well-known sites, as in Lauinger,¹⁵ or if the actions can be circumvented by technologically savvy consumers, as in the case of increased VPN use among individuals affected by the U.K. blocks. In short, the effectiveness of supply-side anti-piracy interventions rests on whether they sufficiently increase consumers' search and transactions costs of finding alternate sources of pirated content.

Conclusion

The literature on copyright enforcement demonstrates that rightsholders have many options for mitigating the impact of piracy on sales, although these strategies often come at a cost to the firms in terms of undermining the effectiveness of their existing marketing strategies.

The literature also shows that government interventions can mitigate the impact of piracy on sales. Although there is little evidence that government action reduces the overall range of content available through piracy channels, there is evidence that government anti-piracy interventions can be effective at changing user behavior if they sufficiently increase the search and transactions costs associated with finding piracy content. Specifically, government actions that only weakly increase the inconvenience of piracy or that are weakly enforced have only transient effects on piracy and legal consumption. However, government actions that meaningfully increase the disutility associated with

illegal filesharing can cause consumers to shift their consumption from illegal channels to legal channels. In light of the empirical evidence, it is clear that anti-piracy actions initiated by both rightsholders and governments can mitigate the economic impact of piracy on legal sales.

However, whether government-sanctioned anti-piracy efforts *can* be effective is a different question from whether government-sanctioned anti-piracy efforts *should* be adopted. The latter involves analyzing the social-welfare impacts of various interventions, as well as a better understanding of the long-run effects of piracy—whether reduced revenues from piracy affect the supply of creative works. As countries continue to evaluate and change their copyright policy over time, and as rightsholders continue to experiment with new strategies in the digital era of copyright, further research is needed to analyze whether these changes affect not just industry revenue but also creative output and social welfare. **□**

Additional background information appears in an online appendix available with this article in the ACM Digital Library; <http://dl.acm.org/citation.cfm?doi=2979673&pi cked=formats>

References

- Adermon, A. and Che-Yuan, L. Piracy and music sales: The effects of an anti-piracy law. *Journal of Economic Behavior and Organization* 105 (Sept. 2014), 90–106.
- Aguilar, L., Claussen, J., and Peukert, C. *Online Copyright Enforcement, Consumer Behavior, and Market Structure*. Working paper. Copenhagen Business School, Copenhagen, Denmark, 2015; <http://ssrn.com/abstract=2604197>
- Christin, N., Weigend, A., and Chuang, J. Content availability, pollution, and poisoning in file-sharing peer-to-peer networks. *Proceedings of the Sixth ACM conference on Electronic Commerce* (Vancouver, B.C., Canada, June 5–8). ACM Press, NY, 2005, 68–77.
- Danaher, B., Smith, M.D., and Telang, R. *The Effect of Piracy Website Blocking on Consumer Behavior*. Working paper. Carnegie Mellon University, Pittsburgh, PA, 2015; <http://ssrn.com/abstract=2612063>
- Danaher, B., Dhanasobhon, S., Smith, M.D., and Telang, R. Understanding media markets in the digital age: Economics and methodology. Chapter 13 in *Economic Analysis of the Digital Economy*, A. Goldfarb, S.M. Greenstein, and C.E. Tucker, Eds. University of Chicago Press, Chicago, IL, 2015, 385–406.
- Danaher, B. and Smith, M.D. Gone in 60 seconds: The impact of the Megaupload shutdown on movie sales. *International Journal of Industrial Organization* 33 (Mar. 2014), 1–8.
- Danaher, B., Smith, M.D., and Telang, R. Piracy and copyright enforcement mechanisms. Chapter 2 in *Innovation Policy and the Economy, Volume 14*, J. Lerner and S. Stern, Eds. National Bureau of Economic Research, University of Chicago Press, Chicago, IL, 2014, 31–67.
- Danaher, B., Smith, M.D., Telang, R., and Chen, S. The effect of graduated response anti-piracy laws on music sales: Evidence from an event study in France. *Journal of Industrial Economics* 62, 3 (Sept. 2014), 541–553.
- Danaher, B. and Waldfogel, J. *Reel Piracy: The Effect of Online Film Piracy on International Box Office Sales*. Working paper. Wellesley College, Wellesley, MA, 2012; <http://ssrn.com/abstract=1986299>
- Danaher, B., Dhanasobhon, S., Smith, M.D., and Telang, R. Converting pirates without cannibalizing

purchasers: The impact of digital distribution on physical sales and Internet piracy. *Marketing Science* 29, 6 (Nov-Dec. 2010), 1138–1151.

- Dey, D., Kim, A., and Lahiri, A. *Online Piracy and the 'Longer Arm' of Enforcement*. Working paper. University of Washington, Seattle, WA, 2015; http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2296116
- Fleming, P., Parravano, M., and Zizzo, D.J. *To Pay or Not to Pay? Determinants of Unlawful Product Acquisition*. CREATE working paper. University of Glasgow, Glasgow, U.K., 2016; <http://www.create.ac.uk/publications/to-pay-or-not-to-pay-determinants-of-unlawful-product-acquisition/>
- Hammond, R.G. Profit leak? Pre-release file sharing and the music industry. *Southern Economic Journal* 81, 2 (Oct. 2014), 387–408.
- Kuczerawy, A. *Private Enforcement of Public Policies: Freedom of Expression in the Era of Online Gatekeeping*. Ph.D. dissertation. Katholieke Universiteit Leuven, Leuven, Belgium, 2016; <https://www.law.kuleuven.be/apps/citip/en/overview/showProject/268/>
- Lauinger, T., Szydłowski, M., Onarlioglu, K., Wondracek, G., Kirida, E., and Kruegel, C. *Clickonomics: Determining the Effect of Anti-Piracy Measures for One-Click Hosting*. Working paper. Northeastern University, Boston, MA, 2013.
- Liebowitz, S. The impacts of Internet piracy. Chapter 13 in *The Economics of Copyright: A Handbook for Students and Teachers*, R. Watt, Ed. Edward Elgar Publishers, Cheltenham Glos, U.K., 2014, 225–240.
- Oberholzer-Gee, F. and Strumpf, K. File sharing and copyright. Chapter 2 in *Innovation Policy and the Economy, Volume 10*, J. Lerner and S. Stern, Eds. National Bureau of Economic Research, University of Chicago Press, Chicago, IL, 2010, 19–55.
- Parloff, R. Megaupload and the twilight of copyright. *Fortune* (July 23, 2012), 21–24; <http://fortune.com/2012/07/11/megaupload-and-the-twilight-of-copyright/>
- Poort, J., Leenheer, J., Ham, J.V.D., and Dumitru, C. Baywatch: Two approaches to measure the effects of blocking access to The Pirate Bay. *Telecommunications Policy* 38, 4 (May 2014), 383–392.
- Reimers, I. Can private copyright protection be effective? Evidence from book publishing. *The Journal of Law and Economics* 59, 2 (May 2016), 411–440.
- Sivan, L., Smith, M.D., and Telang, R. *Do Search Engines Influence Media Piracy? Evidence from a Randomized Field Study*. Working paper. Carnegie Mellon University, Pittsburgh, PA, 2015; <http://ssrn.com/abstract=2495591>
- Smith, M.D. and Telang, R. *Windows of Opportunity: The Impact of Piracy and Delayed International Availability on DVD Sales*. Working paper. Carnegie Mellon University, Pittsburgh, PA, 2015; <http://ssrn.com/abstract=2784759>
- Telang, R. and Waldfogel, J. *Piracy and New Product Creation: A Bollywood Story*. Working paper. Carnegie Mellon University, Pittsburgh, PA, 2014; <http://ssrn.com/abstract=2478755>
- Waldfogel, J. Copyright protection, technological change, and the quality of new products: Evidence from recorded music since Napster. *Journal of Law and Economics* 55, 4 (Nov. 2012), article 1.
- Watson S.J., Zizzo D.J., and Fleming P. Determinants of unlawful file sharing: A scoping review. *PLoS One* 10, 6 (June 2015), e0127921.
- Zhang, L. Intellectual property strategy and the long tail: Evidence from the recorded music industry. *Management Science* (published online in *Articles in Advance*, Nov. 11, 2016); <http://dx.doi.org/10.1287/mnsc.2016.2562>

Brett Danaher (danaher@chapman.edu) is an assistant professor of economics and management science at the Argyros School of Business and Economics, Chapman University, Orange, CA.

Michael D. Smith (mds@cmu.edu) is a professor of information technology and marketing and co-director of the Initiative for Digital Entertainment Analytics at the H. John Heinz III College of Carnegie Mellon University, Pittsburgh, PA.

Rahul Telang (rtelang@andrew.cmu.edu) is a professor of information systems and management and co-director of the Initiative for Digital Entertainment Analytics at the H. John Heinz III College of Carnegie Mellon University, Pittsburgh, PA.

Copyright held by the authors.

DOI:10.1145/2959085

It is past time to acknowledge 400 years of European computational innovation from non-English-speaking scientists and engineers.

BY HERBERT BRUDERER

Computing History Beyond the U.K. and U.S.: Selected Landmarks from Continental Europe

MOST HISTORIES OF computing are dominated by Anglo-Saxon accounts in which devices and practices from elsewhere, continental Europe in particular, are underrepresented and in some cases omitted. However, there is a rich history of such discoveries and the widespread use of computational devices. This article aims to supplement and correct widely accepted accounts, briefly describing examples from European countries in chronological order. Some of these innovations are well known, but, for others, we are no longer aware of them or they are forgotten

entirely. Electronic digital computers abruptly replaced digital mechanical calculating machines and analog logarithmic slide rules in the 1970s. The great calculating machines built by Wilhelm Schickard, Blaise Pascal, and Gottfried Wilhelm Leibniz are not included in this survey.

17th Century

Counting boards. In the early modern period, beautiful counting boards (see Figure 1) were used in many city halls throughout central Europe for addition and subtraction (using counters). Surviving tables (16th to 18th centuries) are today to be found mostly in museums in Switzerland and Germany.¹⁶ Counter reckoning, also called “counter casting” and “calculating on the lines,” was recommended by the abacists and superseded by “pen reckoning” supported by the algorists.

Sectors. There are uncertainties about the origin of the sector (see Figure 2), which was designed in Italy in the 16th century, meaning Galileo Galilei was not its inventor, as is commonly credited. A similar analog instrument is the versatile proportional compass (such as for multiplication, division, and proportion). Sectors were largely abandoned following widespread use of linear slide rules and circular slide rules (both invented by William Oughtred of England).

» key insights

- Spanish engineer Leonardo Torres Quevedo built two sophisticated, fully operational endgame-chess-playing machines in the early 20th century, showing that “artificial intelligence” began decades before Alan Turing and Konrad Zuse published their research.
- Alan Turing’s 1936 paper on the universal Turing machine was still almost unknown at a major conference on calculating machines and human thinking in Paris in 1951.
- The French clockmaker Jean-Baptiste Schwilgué designed a mechanical adder to “control” a milling machine for manufacturing precision gears; his key-driven adder (1844) predates Du Bois D. Parmelee’s device (1850).



Figure 1. A rare counting board (this one from the 16th century), once very common in the town halls of central Europe; to perform calculations a user would need to put tokens, or *rechenpfennige*, on the lines (=value 1, 10, 100, 1,000) or between them (=value 5, 50, 500, 5,000) of at most four pieces in the same place. Courtesy of Historisches Museum, Basel, Switzerland.



Figure 3. Early programmable handwriting automaton with internal mechanics built by Pierre Jaquet-Droz, 18th century; the text (up to 37 characters) is stored on cam plates. Courtesy of Musée d'Art et d'Histoire, Neuchâtel, Switzerland.



Figure 2. Sector, a universal calculating instrument developed in the 16th century containing various scales (such as trigonometric) depending on scope; a pair of dividers is necessary for multiplication or division. The sectors are based on the Thales' intercept theorem on intersecting lines.^{1,25} Courtesy of the Collection of Astronomical Instruments, ETH Library Zürich, Switzerland.



Figure 4. Early key-driven adding machine, or so-called "direct adding machine," developed by Jean-Baptiste Schwilgué of Strasbourg, patented in 1844; pressing a key causes the corresponding number to be stored in the register, making addition very quick. Courtesy of the Collection of Astronomical Instruments, ETH Library Zürich, Switzerland.

18th Century

Programmable handwriting automaton.

Friedrich Knaus of Germany in 1760 constructed a marvelous programmable automatic handwriting machine. His "Alles schreibende Wundermaschine" is today on display at the Technisches Museum in Vienna. The Swiss watchmaker Pierre Jaquet-Droz created in 1772 his famous *écrivain*, or writer (see Figure 3), now on display at the Musée d'art et d'histoire, Neuchâtel, Switzerland, a machine that is still operational. In both cases the user

may input a short text with some limitations on capital letters (Wundermaschine 68 characters, *écrivain* 37 characters). The complex mechanism is either outside (Vienna) or inside (Neuchâtel). The texts are written in ink with a pen.^{20,34}

The most famous forerunner of Jaquet-Droz was Jacques Vaucanson of France. Unfortunately, his duck-flute-drum player automata were destroyed. Another important maker of automata was Peter Kintzing of Germany; for details see Bruderer and Meilensteine.⁶

19th Century

Early key-driven adder.

To my knowledge no book on the history of computing mentions the world-famous watchmaker Jean-Baptiste Schwilgué of France, creator of the astronomical clock in the Strasbourg Cathedral. He received a patent in 1844 for his key-driven adding machine (see Figure 4). One copy (1846) is today in the Musée historique of Strasbourg, the other (1851) at ETH Zürich. The Swiss machine is in much better condition. There was an earlier Italian key-driven

calculation machine (1834) developed by Luigi Torchi of Italy, but little is known about it today.

In general, authors writing about the history of computing regard the device (1850) of Du Bois D. Parmelee of the U.S. as the “first” key-driven adder,³³ sometimes citing the Schilt machine (1850) by Victor Schilt, a Swiss watchmaker from Solothurn who had worked with Schwilgué. This calculation aid, now in the collection of National Museum of American History in Washington, D.C., was shown in 1851 at the Great Exhibition in London. However, the leading publications on computer history do not mention Schwilgué’s key-driven adder. It is not known how many cop-

ies of this single-digit adder were built, though many multiple-order key-driven machines were in the U.S. by the end of the 19th century.

Early “process computer.” For the construction of his splendid Strasbourg astronomical clock, Schwilgué developed several complicated machines, including a very precise milling machine for producing complex gears (circa 1827) and a large, specialized calculating machine (circa 1830); for dating and technical details, see Bruderer and Meilensteine.⁶ This device, which is driven by a crank and a weight, helped Schwilgué compute the values needed for the settings of his milling machine. He manually transferred the calculations to

a paper tape he would then place in a box beside the milling machine. These numbers controlled the machine. It might thus be considered a simple “process computer” (or precursor). As far as is known, Schwilgué constructed only one such highly specialized machine.

Several books and papers were published by Schwilgué’s collaborators and successors (foremost Alfred Ungerer of France) with a short description and picture of the milling machine. The machine is also mentioned in a biography of Schwilgué’s son, Charles.

To my surprise I came across Schwilgué’s adding machine (see Figure 5) in December 2014 in Strasbourg. Both devices are today in the Musée historique de Strasbourg.

Other historic calculating devices include the common slide adders (see Figure 6). These inexpensive, mass-produced instruments were manu-

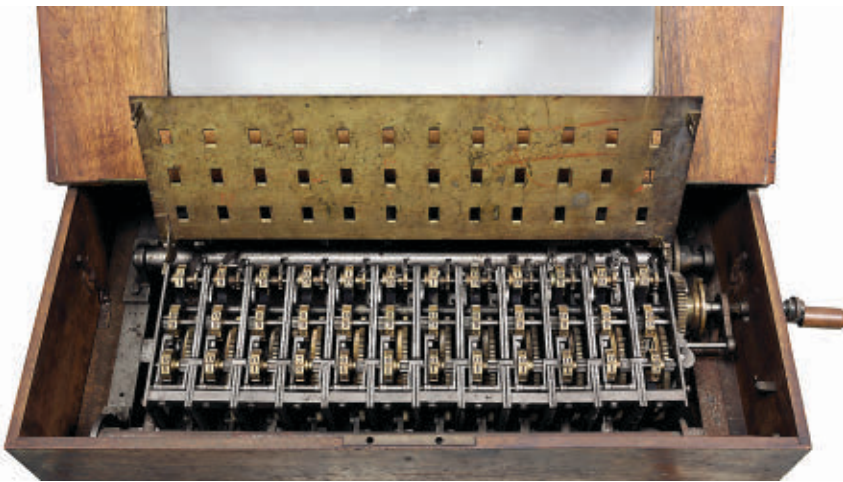


Figure 5. Schwilgué’s large adding machine that probably facilitated the calculations needed to produce the complex gears of the astronomical clock of Strasbourg Cathedral; the original weight drive is lost. Courtesy of Mathieu Bertola, Musée historique, Strasbourg, France.

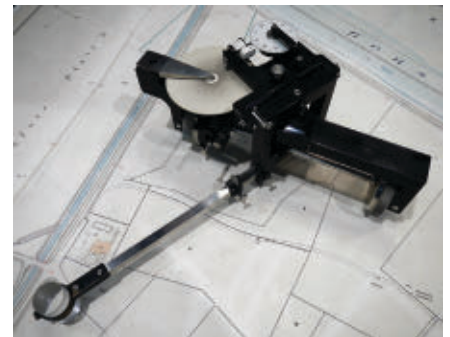


Figure 7. Polar planimeter invented by Jakob Amsler in 1854 was used to calculate surfaces by applying differential and integral calculus. Only one manufacturer remains active today, Gebrüder Haff GmbH, Pfronten, Germany. Courtesy of Amt für Vermessung, Aarau, Switzerland.

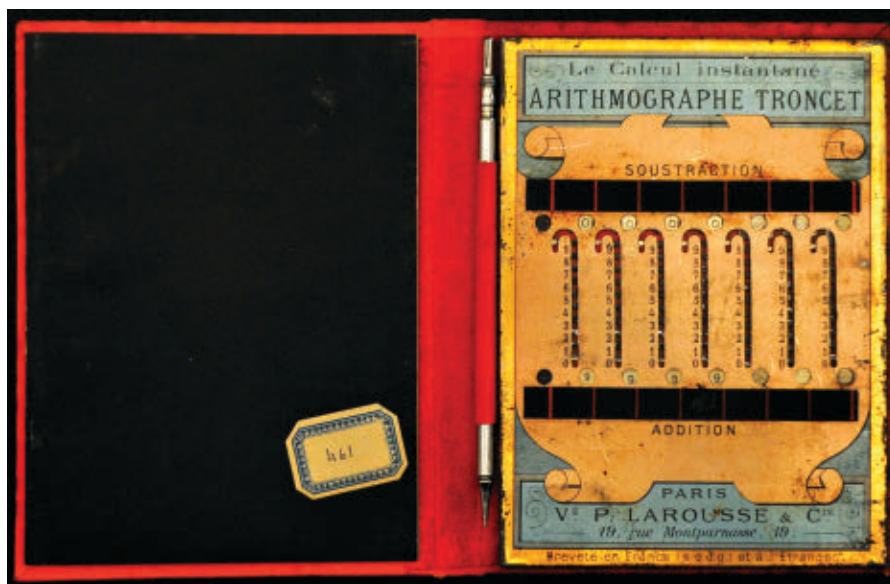


Figure 6. Slide adder devised by the French manufacturer Louis Troncet in 1889; the tens carry is semiautomatic, needing a mechanism that looks like a cane. Such an instrument is also able to perform subtractions and is sometimes combined with multiplication tables. Courtesy of Herbert Spühler, Stallikon, Switzerland.



Figure 8. Music boxes are still produced by the Swiss firm Reuge and often presented as gifts; the melodies were originally stored on (exchangeable) pegged brass cylinders, though the cylinders were later replaced by (cheaper) perforated disks. Courtesy of Reuge SA, Sainte-Croix, Switzerland.

factured in Germany, France, and Switzerland and widely disseminated. Credit is generally attributed to Heinrich Kummer of Germany (1847).

The “planimeter” was invented at the beginning of the 19th century by Johann Martin Herrmann of Germany (1814), Tito Gonnella of Italy (1824), and Johannes Oppikofer of Switzerland (1827). Much more successful were the precise polar planimeters (see Figure 7). The three most influential early designers and manufacturers of these and other integrating instruments were Jakob and Alfred Amsler of Schaffhausen, Switzerland, Gottlieb Coradi of Zürich, and Albert Ott of Kempten, Germany.

The Swiss cylinder musical box (see Figure 8) uses a sophisticated program store (pinned cylinder). Its successor was the German disk musical box with a perforated disk (invented in 1885) that involved much simpler production. The many mechanical musical instruments in Europe at the time were replaced by the phonograph (Emile Berliner of Germany and the U.S.) and the gramophone (Thomas Alva Edison and his Swiss-born engineer Johann Heinrich Krüsi). Another

form of storage was music rolls (perforated paper rolls).

20th Century

Cylindrical slide rule. The world’s largest and most precise mass-produced cylindrical slide rule (see Figure 9) was manufactured by Loga Calculator AG in Zürich and Uster, Switzerland. The drum contains 80 sections, each 60 centimeters long; the length of the scale (due to overlapping) is 24 meters. Prior to mid-2013, only three surviving copies were known. Since then, four more have been discovered in Switzerland. Fuller’s spiral slide rule has a scale of 12.7 meters; for details see Bruderer and Meilensteine.⁶

Chess automatons. Playing chess is often regarded as requiring intelligence. Two operational chess automata were created in Spain at the beginning of the 20th century by Spanish engineer Leonardo Torres (y) Quevedo, who also built a cable car for spanning a portion of Niagara Falls.

In 1912, he designed his first electromechanical chess machine (see Figure 10), followed by a second device several years later (see Figure 11). Both machines are today on display

at the Museo Leonardo Torres Quevedo in Madrid. Austrian computer pioneer Heinz Zemanek saw the chess automaton demonstrated at the World Exhibition in Brussels in 1958. These devices, which did not play a complete chess game, were restricted to the end game—king and rook against king.



Figure 10. Early electromechanical chess automaton (1912) by Torres Quevedo; unlike Wolfgang von Kempelen’s machine, it is an authentic automaton without hidden human chess player. Courtesy of Museo Leonardo Torres Quevedo, Universidad Politécnica de Madrid, Spain.



Figure 9. The world’s largest mass-produced cylindrical slide rule from Loga Calculator (circa 1912), Zürich/Uster, Switzerland; length of scale: 24 meters. Multiplication is reduced to addition and division to subtraction in the same way traditional slide rules work. These devices were common in banks and insurance companies worldwide. Courtesy of UBS, Basel, Switzerland.



Figure 11. Second automatic chess endgame machine by Torres Quevedo, playing with king and rook (automaton) vs. king (human player). Courtesy of Museo Leonardo Torres Quevedo, Universidad Politécnica de Madrid, Spain.



Figure 12. Torres Quevedo's electromechanical arithmometer (1920), typewriter controlled, with conditional branching, based on Charles Babbage's analytical engine. Courtesy of Museo Leonardo Torres Quevedo, Universidad Politécnica de Madrid, Spain.

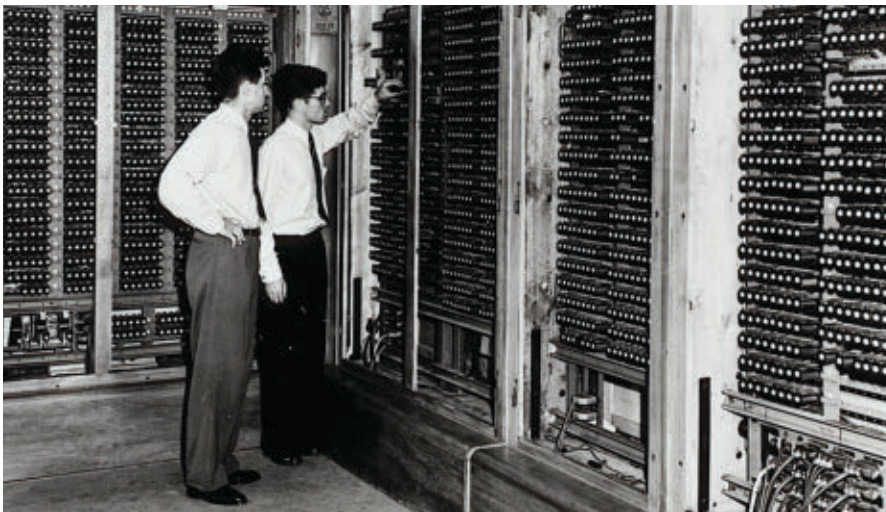


Figure 13. Zuse's binary relay computer Z4 was first commercially available in 1945; this tape-controlled programmable machine with floating-point arithmetic was in operation in Zürich from 1950 to 1955 and included an innovative mechanical memory without relays. Courtesy of ETH Library Zürich, Switzerland.



Figure 14. Grande Dixence in the Swiss Alps, the world's highest concrete dam, relied on calculations aided by the Zuse Z4 and electromechanical desktop calculators (such as Madas from H. W. Egli AG, Zürich-Wollishofen). Courtesy of Grande Dixence SA, Sion, Switzerland.

These two sophisticated “intelligent” chess machines were built approximately 30 years before Alan Turing of the U.K. and Konrad Zuse of Germany first thought about computer chess.

Electromechanical “analytical engine.” Torres Quevedo also tried to build an analytical engine (see Figure 12) controlled by a remote typewriter and incorporating several notable features of conditional branching, presenting it in Paris in 1920. He also published an important theoretical paper on floating point arithmetic.³²

Early commercially available computer. Many historians view the Ferranti Mark 1 (in the U.K.) and the Univac (in the U.S.) as the “first” commercially available computers, both delivered in 1951. However, the German relay calculator Zuse Z4 (see Figure 13) was already operational in 1945, with the ETH Zürich renting it in 1949. It remained in operation in Switzerland from 1950 to 1955 and is today on display at the Deutsches Museum in Munich.

The Zuse Z4 was used in Zürich for scientific and industrial purposes. Two applications were the tension calculations for the Grande Dixence dam (world's highest) in the Canton of Valais, Switzerland (see Figure 14) and flutter calculations for the jet fighter P-16 of Flug- und Fahrzeugwerke Altenrhein AG, St. Gallen, Switzerland (see Figure 15).⁵

Relay calculator Bark. Several computer scientists view the tape-controlled Zuse Z4 as the only functioning computer in continental Europe in 1950. Yet there was another relay machine in Stockholm, Sweden, where Bark was controlled via plug board and was in operation until 1955 before being dismantled.²⁷

Early programming language. Plankalkül developed by Zuse (1945) is regarded as one of the earliest programming languages. Zuse also anticipated chess programming.⁴¹

Automatic programming. Donald Knuth²¹ considers mathematician Heinz Rutishauser of Switzerland a father of automatic programming. In 1951, Rutishauser suggested using the computer itself to write programs, publishing “Automatische Rechenplanfertigung” (automatic production of programs) in 1952.³⁰ These efforts later led to the programming language Algol. Though many historians view Grace

Hopper as the “mother” of the compiler, Donald Knuth says that Alick Glennie of Manchester, U.K., should share credit for this achievement.²¹

Calculating punch M9. In the 1950s, Zuse manufactured a series of more than 20 calculating punches for Remington Rand in Zürich. I rediscovered in 2011 one of the M9s (see Figure 16), which is today at the Museum für Kommunikation in Berne, Switzerland.

Böhm’s compiler. Pioneer Corrado Böhm of Italy published his doctoral thesis at ETH Zürich in 1954, writing a compiler in its own language.²¹ He had been, 1949–1950, a member of Eduard Stiefel’s staff at the Institute for Applied Mathematics in Zürich. Along with engineer Harry Laett he tested the legendary relay calculator Zuse Z4 in 1949 prior to its installation in Zürich.⁵ The “meta-circular compiler” mentioned as part of the Corrado Böhm biography at http://www.corradobohm.it/Corrado_Bohm/Biography.html is the first known example of such a compiler.

Transistorized computer Mailüfterl. One of the earliest European transistorized computers was built by pioneer Heinz Zemanek of Austria in 1958. Called Mailüfterl, or “weak spring wind” (after the large MIT computer Whirlwind),⁴⁰ it is today on display at the Technisches Museum in Vienna.

Transistorized computer Cora. Researchers at the Ecole polytechnique fédérale Lausanne (EPFL) in 2011 publicly credited Hungarian engineer Peter Tóth with designing the first known Swiss transistorized computer. The only preserved Cora (see Figure 17) is today on display at the EPFL.^{5,6}

Ultimate mechanical pocket calculators/smallest mechanical parallel calculator. From 1949 to 1971 engineer Curt Herzstark of Austria working in Liechtenstein produced two magnificent pocket calculating machines both called “Curta” (see Figure 18). Approximately 130,000 were manufactured and sold worldwide during that time.

In November 2015 I found at Schreibmaschinenmuseum Beck, Pfäffikon, Switzerland, original engineering drawings and patent documents detailing an unknown multiple Curta (see Figure 19),^{18,19} generally believed to be the world’s smallest mechanical parallel calculator. Two, four, or five con-



Figure 15. Swiss jet fighter P-16 at airport in Flug- und Fahrzeugwerke AG, Altenrhein, Switzerland, on Lake Constance near the German border; flutter calculations were aided by the Zuse Z4 for this supersonic plane in the 1950s. Courtesy of Staatsarchiv, St. Gallen, Switzerland.

ventional Curtas are combined, with one single crank needed to operate the combined machines. For more, see the newsletter (Spring 2016) of the Charles Babbage Institute (<http://www.cbi.umn.edu/about/nsl/v38n1.pdf>) and the journal *Resurrection* (Autumn 2016) of the Computer Conservation Society (<http://www.computerconservation-society.org/resurrection/pdfs/res75.pdf>).

Building an Electronic Digital Computer

After World War II many universities in Europe and elsewhere sought to build



Figure 16. Zuse’s program-controlled parallel decimal electromechanical “calculating punch” M9 manufactured for Remington Rand, Zürich; combined with punched-card machines, the M9 was used for multiple applications (such as for accountancy and statistics). Courtesy of Max Forrer, Oberhelfenschwil, Switzerland.



Figure 17. Swiss transistorized computer Cora developed and manufactured by Contraves AG, Zürich, in 1963 originally for military purposes as a fire-control calculator. Courtesy of Musée Bolo, Ecole polytechnique fédérale Lausanne, Switzerland.

their own electronic digital computers. For a number of reasons, including lower cost, independence from foreign countries, education of home-grown mathematicians and engineers, and adaptation to own needs, they often preferred to design the machines themselves rather than buy them on the worldwide market. But how to acquire the necessary knowhow? At the time, the relatively few books and courses on the subject were largely unavailable. Many British and continental European mathematicians and engineers thus spent time in the U.S. at such institutions as the Institute for Advanced Study at Princeton (under the direction of John von Neumann) or at Harvard University (under the direction of Howard Aiken).⁸

Swiss pioneers Eduard Stiefel, Heinz Rutishauser, and Ambros Speiser in 1949–1950 spent several months at Harvard and Princeton, as well as in the U.K. The result of their investigations was a fundamental book about computers called *Programmgesteuerte digitale Rechengeräte*. Stiefel had previously, in 1948, founded the Institute for Applied Mathematics of ETH Zürich.

Book on the Building of Stored Program Computers

As investigated by Arnold Cohen,¹³ there were, at the beginning of the 1950s, two main books on the construction of stored-program computers: *High-speed Computing Devices* by Engineering Research Associates (1950)¹³ and *Programmgesteuerte digitale Rechengeräte* (Program-Controlled Electronic Digital Computers) by Rutishauser et al.³¹ This work was published in four parts, 1950–1951, in the German scientific journal *Zeitschrift für angewandte Mathematik und Physik* and as a book in 1951 (see Figure 20). It includes a worldwide overview of then-current computing machines and projects. The authors discussed such topics as the advantages and disadvantages of serial and parallel processing, fixed and floating point arithmetic, conditional branching, program storage, and self-modifiable programs.

Heinz Zemanek³⁹ wrote that the report by Rutishauser, Speiser, and Stiefel was “jahrelang die beste Dokumentation über den Computer in deutscher Sprache” (“the best documentation for many years on electronic digital computers in the German language”).

Paris Computer Conference of 1951

The conference included 268 participants, among them 10 women, mostly “calculatrices,” or female computers, from 10 countries and covered calculating machines and human thinking. This was probably the most important early computer congress in Europe; an earlier meeting had been held in Cambridge, U.K., in 1949.

Many European and American pioneers attended, including Howard Aiken (Cambridge, MA), Ross Ashby (Gloucester, U.K), Andrew Booth (London, U.K.), Bertram Bowden (Manchester, U.K.), Francis Colebrook (Teddington, U.K.), Louis Couffignal (Paris, France), Douglas Hartree (Cambridge, U.K.), Tom Kilburn (Manchester, U.K.), Göran Kjellberg (Stockholm, Sweden), Warren McCulloch (Chicago, IL), Conny Palm (Stockholm, Sweden), Mauro Picone (Rome, Italy), Eduard Stiefel (Zürich, Switzerland), Gonzales Torres Quevedo (Madrid, Spain), Albert Uttley (Great Malvern, U.K.), Willem van der Poel (The Hague, the Netherlands), Adriaan van Wijngaarden (Amsterdam, the Netherlands), Grey Walter (Bristol, U.K.), Alwin Walther (Darmstadt, Germany), Norbert Wiener (Cambridge, MA), Mau-



Figure 18. Curta, the world’s smallest mechanical pocket calculator, is a stepped drum machine able to perform all four basic arithmetic operations; Curt Herzstark, deported from Austria by the Nazis in 1943, was compelled to design it while imprisoned in the Buchenwald concentration camp. Courtesy of Sven Beham, Liechtensteinisches Landesmuseum, Vaduz, Liechtenstein.

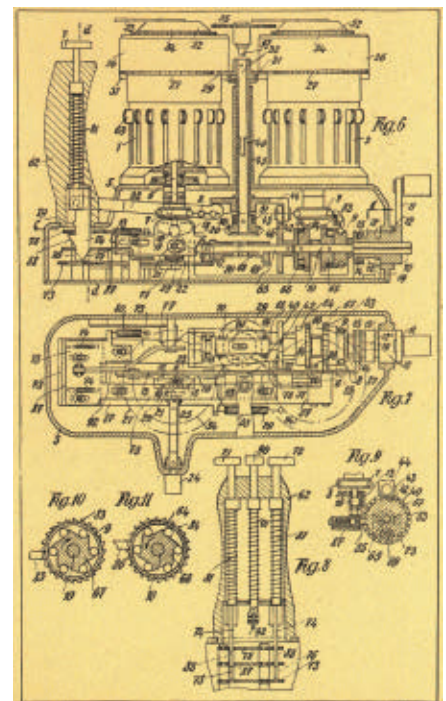


Figure 19. This engineering drawing of the Double Curta illustrates one of the four arrangements Herzstark proposed—horizontal duplex mechanical pocket calculator. Courtesy of Schreibmaschinenmuseum Beck, Pfäffikon, Switzerland.

rice Wilkes (Cambridge, U.K.), Frederic Williams (Manchester, U.K.), and John Womersley (Letchworth, U.K.). Alan Turing did not participate. All papers were translated into French. Norbert Wiener played against the second version of Torres Quevedo's chess automaton (operated by his son Gonzales). In his paper, Francis Colebrook (officer-in-charge, Electronics Section, National Physical Laboratory, Teddington, U.K.) referred to Turing's abstract treatise on the universal machine (1936) but did not mention the concept of a stored program.

The conference is not well known today since its 589-page proceedings²⁷ is still available only in French (see Figure 21). It seems to be one of the earliest large gatherings to explore the themes of human thought and computing machines. In 1956, the Dartmouth summer research project on artificial intelligence took place in Hanover, NH.

Priority and Patriotism

There are still endless debates over such questions as “Who discovered the logarithms?,” “Who invented the computer?,” “Who created the stored program?,” and “Who is the father of artificial intelligence?” Answers can vary

Gone today, however, are the world's former leading makers of mechanical integrators, including Amsler (Schaffhausen, Switzerland), Coradi (Zürich, Switzerland), and Ott (Kempten, Germany).

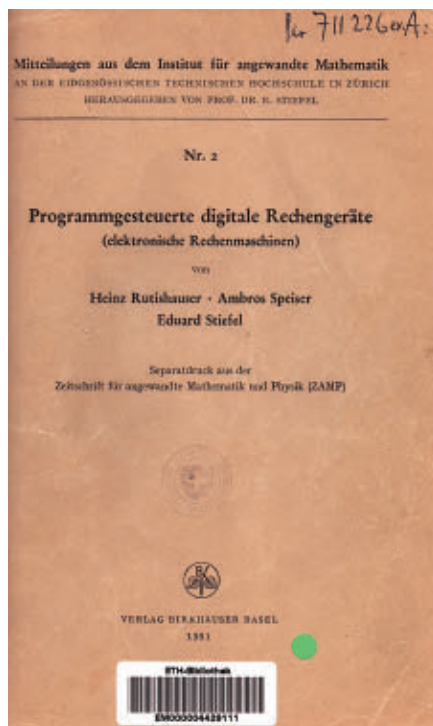


Figure 20. One of the most influential early books from continental Europe on the building of electronic stored-program computers includes no mention of the universal Turing machine. Courtesy of ETH Library Zürich, Switzerland.

depending on nationality of the person answering. Germans are most likely to favor Konrad Zuse and Kurt Gödel, the British Alan Turing, and the Americans and Hungarians John von Neumann.

As this article focuses on computing history outside the U.K. and the U.S., I do not discuss this matter in detail; for more, see selected contributions by Maston Beard,² Allan Bromley,⁴ Maarten Bullynck,¹⁰ Jack Copeland,¹¹ Edgar Daylight,¹² Thomas Haigh,^{14,15} Allan Olley,²⁴ Eloína Peláez,²⁶ and Mark Priestley.²⁸

Several independent inventors were also involved in both the mechanical precursors and the electronic digital computer. Who was “first” depends on one's definition of “computer.”^{37,38} For example, it is likely there were also several creators of the stored program concept. Notably, Presper Eckert, John Mauchly, and John von Neumann (all of the U.S.) had to overcome technical bottlenecks. Konrad Zuse in 1936 wrote in a patent application about the possibility of internal memory.^{5,11,41} Meanwhile, many terms have since changed meaning; for example, until the 1940s, “computers” were human beings doing calculations, usually with the help of mechanical calculating machines.

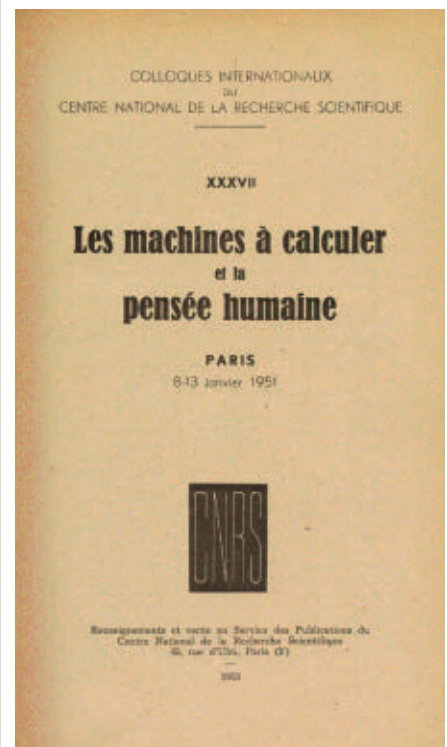


Figure 21. The voluminous French-language proceedings of the 1951 conference on computing machines and human thinking. Courtesy of ETH Library Zürich, Switzerland.


Conclusion

The leading scientific journal worldwide in the post-war period was *Mathematical Tables and Other Aids to Computation*, first published by the American Mathematical Society in 1943. From 1954 to 1957, the *Digital Computer Newsletter* was published by the Office of Naval Research within the Navy Department in Washington, D.C., and as a supplement to the *Journal of the Association for Computing Machinery*. Both covered computer developments outside the U.S. and the U.K. *Mathematical Tables and Other Aids to Computation* included many reviews of non-English works in computing research. Unfortunately, many current U.S. and British books and journals on the history of computing do not adequately acknowledge the contributions of non-English-speaking countries.

For example, there were two co-discoverers of logarithms—Jost Bürgi of Switzerland and John Napier of Scotland. Bürgi developed the logarithms first, and Napier published his results first.³⁵ The invention of the pantograph is generally attributed to Christoph Scheiner of Germany (1603), but Heron of Alexandria (first century) should be credited for (a different type of) this drawing instrument.^{3,17,23}

Electronic devices quickly replaced mechanical machines and instruments in the 1970s. Inventions from continental Europe, including sectors, proportional compasses, planimeters, and pantographs disappeared worldwide. Gone today, however, are the world's former leading makers of mechanical integrators, including Amsler (Schaffhausen, Switzerland), Coradi (Zürich, Switzerland), and Ott (Kempten, Germany). Forgotten are the mechanical and electronic analog computers produced in Germany (such as Telefunken) and Switzerland (Amsler, Contraves, and Güttinger). Including these significant achievements in non-English-speaking countries enriches the history of computing for all.

Acknowledgment

I am very grateful to Thomas J. Misa of the Charles Babbage Institute at the University of Minnesota for copy editing this article and to Thierry Amstutz and Christian Hörack of the Musée d'art et d'histoire, Neuchâtel, Switzerland, for the Jaquet-Droz automata. 

Further reading

This article is based in part on my 2015 book *Milestones in Analog and Digital Computing*, which includes a comprehensive bibliography.⁶

References

1. Agriola, I. and Friedrich, T. *Elementary geometry*. American Mathematical Society, Providence, RI, 2008, 9–11.
2. Beard, M. and Pearcey, T. The genesis of an early stored-program computer: Csirac. *Annals of the History of Computing* 6, 2 (Apr.–June 1984), 106–115.
3. Beck, Th. Herons des Älteren Mechanik. *Beiträge zur Geschichte der Technik und Industrie I* (1909), 87–88.
4. Bromley, A. The origin of the stored program concept. In *The last of the first, Csirac: Australia's First Computer*, D. McCann and P. Thorne, Eds. Department of Computer Science and Software Engineering, University of Melbourne, Melbourne, Australia, 2000, 87–95.
5. Bruderer, H. *Konrad Zuse und die Schweiz. Wer erfand den Computer?* (Konrad Zuse and Switzerland. Who invented the Computer?). De Gruyter Oldenbourg, Munich/Berlin, 2012.
6. Bruderer, H. *Meilensteine der Rechentechnik. Zur Geschichte der Mathematik und der Informatik* (Milestones in Analog and Digital Computing. Contributions to the History of Mathematics and Information Technology). De Gruyter Oldenbourg, Berlin/Boston, 2015.
7. Bruderer, H. The birth of artificial intelligence. First conference on artificial intelligence in Paris in 1951? In *International Communities of Invention and Innovation. History of Computing 2016, International Federation for Information Processing Advances in Information and Communication Technology 491*. A. Tatnall and C. Leslie, Eds. Springer International Publishing AG, Cham, Switzerland, 2016, 181–185.
8. Bruderer, H. The world's smallest mechanical parallel calculator. Discovery of original drawings and patent documents from the 1950s in Switzerland. In *International Communities of Invention and Innovation. History of Computing 2016, International Federation for Information Processing Advances in Information and Communication Technology 491*. A. Tatnall and C. Leslie, Eds. Springer International Publishing AG, Cham, Switzerland, 2016, 186–192.
9. Bruderer, H. Early history of computing in Switzerland: Discovery of rare devices, unknown documents, and scarcely known facts. *IEEE Annals of the History of Computing* 37, 1 (Jan.–Mar. 2017).
10. Bullynck, M., Daylight, E.D., and De Mol, L. Why did computer science make a hero out of Turing? *Commun. ACM* 58, 3 (Mar. 2015), 37–39.
11. Copeland, J. and Sommaruga, G. The stored-program universal computer: Did Zuse anticipate Turing and von Neumann? In *Turing's Revolution. The Impact of His Ideas About Computability*, G. Sommaruga and T. Strahm, Eds. Birkhäuser/Springer, Cham, Switzerland, 2015, 43–101.
12. Daylight, E. *The Dawn of Software Engineering. From Turing to Dijkstra*. Lonely Scholar, Heverlee, Belgium, 2012.
13. Engineering Research Associates, Inc. *High-Speed Computing Devices*. Tomash Publishers, Los Angeles, San Francisco, 1983 (first published in 1950).
14. Haigh, T. Actually, Turing did not invent the computer. *Commun. ACM* 57, 1 (Jan. 2014), 36–41.
15. Haigh, T., Priestley, M., and Rope, C. Reconsidering the stored-program concept. *IEEE Annals of the History of Computing* 36, 1 (Jan.–Mar. 2014), 4–17.
16. Hergenahn, R., Reich, U., and Rochhaus, P. *Mache für dich Linien ... Katalog der erhaltenen originalen Rechenische, Rechenbretter und -tücher der frühen Neuzeit*. Adam-Ries-Bund, Annaberg-Buchholz, Germany, 1999.
17. Héron d'Alexandrie. *Les mécaniques ou l'élevateur des corps lourds*. Société d'édition Les belles lettres. Paris, France, 1988, 228–231.
18. Herzstark, C. Austrian Patent No. 195 147; basic patent: Jan. 25, 1958; patent application: Oct. 19, 1954.
19. Herzstark, C. Austrian Patent No. 205 775; additional patent: Oct. 10, 1959; patent application: Dec. 15, 1954.
20. Kang, M. *Sublime Dreams of Living Machines: The Automaton in the European Imagination*. Harvard University Press, Cambridge, MA, 2011.
21. Knuth, D.E. and Trabb PL. The early development of programming languages. In *A History of Computing in the Twentieth Century. A Collection of Essays*, N.C.

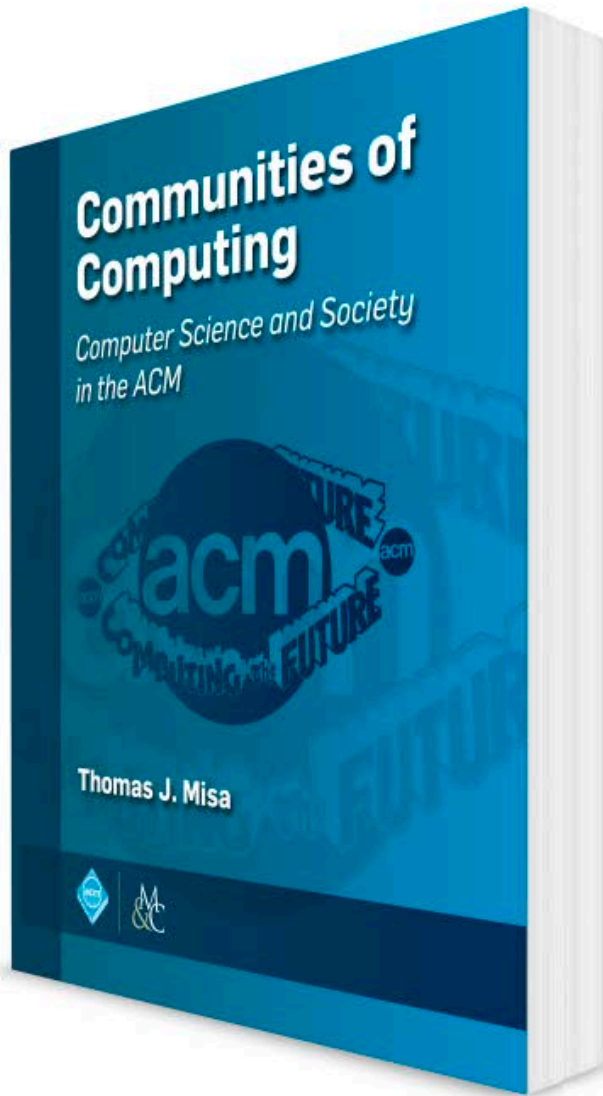
- Metropolis, J. Howlett, and G.-C. Rota, Eds. Academic Press, New York, 1980, 197–273.
22. Martin, E. *The Calculating Machines (Die Rechenmaschinen). Their History and Development*. MIT Press, Cambridge, MA, London/Tomash publishers, Los Angeles, San Francisco, 1992.
23. Nix, L.M.L. and Schmidt, W., Eds. *Hérons von Alexandria Mechanik und Katoptrik*. B.G. Teubner, Leipzig, Germany, 1900, 30–34.
24. Olley, A. Existence precedes essence: Meaning of the stored-program concept. In *History of Computing. Learning from the Past. International Federation for Information Processing Working Group 9.7 International Conference, History of Computing 2010*, A. Tatnall, Ed. (Brisbane, Australia, Sept. 20–23), Springer, Berlin, Germany, 2010, 169–178.
25. Ostermann, A. and Wanner, G. *Geometry By Its History*. Springer-Verlag, Berlin, Heidelberg, Germany, 2012, 3–5.
26. Peláez, E. The stored-program computer: Two conceptions. *Social Studies of Science* 29, 3 (May–June 1999), 359–389.
27. Pèrès, J., Ed. *Les machines à calculer et la pensée humaine. Colloques internationaux du Centre national de la recherche scientifique, Nr. 37* (Paris, France, Jan. 8–13). Editions du Centre national de la recherche scientifique (CNRS), Paris, France, 1953.
28. Priestley, M. *A Science of Operations. Machines, Logic and the Invention of Programming*. Springer-Verlag, London, U.K., 2011.
29. Randell, B., Ed. *The Origins of Digital Computers. Selected Papers, Third Edition*. Springer-Verlag, Berlin, Heidelberg, Germany, 1982.
30. Rutishauser, H. *Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen*. Birkhäuser Verlag, Basel, Switzerland, 1952.
31. Rutishauser, H., Speiser, A., and Stiefel, E. *Programmgesteuerte digitale Rechengerate (Elektronische Rechenmaschinen)*. Birkhäuser Verlag, Basel, Switzerland, 1951.
32. Torres Quevedo, L. Ensayos sobre automática. Su definición. Extensión teórica de sus aplicaciones. *Revista de la real academia de ciencias exactas, físicas y naturales* 12 (1913/14), 391–419.
33. Turck, J.A.V. *Origin of Modern Calculating Machines. A Chronicle of the Evolution of the Principles that Form the Generic Make-Up of the Modern Calculating Machine*. The Western Society of Engineers, Chicago, IL, 1921.
34. Voskuhl, A. *Androids in the Enlightenment: Mechanics, Artisans, and Cultures of the Self*. University of Chicago Press, Chicago, IL, 2013.
35. Waldvogel, J. Jost Bürgi and the discovery of the logarithms. *Elemente der Mathematik* 69, 3 (July–Sept. 2014), 89–117.
36. Williams, M.R. *A History of Computing Technology*. IEEE Computer Society Press, Los Alamitos, CA, 1997.
37. Williams, M.R. What does it mean to be the first computer? In *Proceedings of the IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing* (Sofia, Bulgaria, Oct. 3–6). IEEE Computer Society, Washington, D.C., 2006, 3–9.
38. Williams, M.R. The first computer. In *New Results and New Trends in Computer Science*, H.A. Maurer, Ed. Springer-Verlag, London, U.K., 1991, 371–387.
39. Zemanek, H. *Weltmacht Computer. Weltreich der Information*. Bechtle-Verlag, Esslingen, München, Germany, 1991, 210.
40. Zemanek, H. Central European prehistory of computing. In *A History of Computing in the Twentieth Century. A Collection of Essays*, N.C. Metropolis, J. Howlett, and G.-C. Rota, Eds. Academic Press, New York, London, 1980, 587–609.
41. Zuse, K. *Der Computer—Mein Lebenswerk, Fifth Edition*. Springer-Verlag, Berlin, Heidelberg, Germany, 2010.

Herbert Bruderer (bruderer@retired.ethz.ch; herbert.bruderer@bluewin.ch) is a retired lecturer in didactics of computer science at ETH Zürich; more recently, he has been an historian of technology and was co-organizer of the International Turing Conference at ETH Zürich in 2012.

Copyright held by the author.
Publication rights licensed to ACM. \$15.00



Watch the author discuss his work in this exclusive *Communications* video. <http://cacm.acm.org/videos/computing-history-beyond-the-uk-and-us>



**Your first book-length
history of the ACM.**

**Defining the Discipline
Broadening the Profession
Expanding Research Frontiers**

Thomas J. Misa (Editor)

Charles Babbage Institute (University of Minnesota)

The SIGs, active chapters, individual members, notable leaders, social and political issues, international issues, computing and community education...all are topics found within this first book-length history of the Association for Computing Machinery (ACM). Featuring insightful profiles of people who shaped ACM, such as Edmund Berkeley, George Forsythe, Jean Sammet, Peter Denning, and Kelly Gotlieb, and honest assessments of controversial episodes, this volume deals with compelling and complex issues involving ACM and computing.

This is not a narrow organizational history. While much information about the SIGs and committees are presented, this book is about how the ACM defined the discipline, broadened the profession, and how it has expanded research frontiers. It is a permanent contribution to documenting the history of ACM and understanding its central role in the history of computing.



ISBN: 978-1-970001-84-6 DOI: 10.1145/2973856

<http://books.acm.org>

<http://www.morganclaypoolpublishers.com/misa>

Model learning emerges as an effective method for black-box state machine models of hardware and software components.

BY FRITS VAANDRAGER

Model Learning

WE ROUTINELY MANAGE to learn the behavior of a device or computer program by just pressing buttons and observing the resulting behavior. Especially children are very good at this and know exactly how to use a smartphone or microwave oven without ever consulting a manual. In such situations, we construct a mental model or state diagram of the device: through experiments we determine in which global states the device can be and which state transitions and outputs occur in response to which inputs. This article is about the design and application of algorithms that perform this task automatically.

There are numerous approaches where models of software components are inferred through analysis of the code, mining of system logs, or by performing

tests. Many different types of models are inferred, for example, hidden Markov models, relations between variables, and class diagrams. In this article, we focus on one specific type of models, namely *state diagrams*, which are crucial for understanding the behavior of many software systems, such as (security and network) protocols and embedded control software. Model inference techniques can be either white box or black box, depending on whether they need access to the code. In this article, we discuss *black box* techniques. Advantages of these techniques are that they are relatively easy to use and can also be applied in situations where we do not have access to the code or to adequate white box tools. As a final restriction, we only consider techniques for *active learning*, that is, techniques that accomplish their task by actively doing experiments (tests) on the software. There is also an extensive body of work on passive learning, where models are constructed from (sets of) runs of the software. An advantage of active learning is that it provides models of the full behavior of a software component, and not just of the specific runs that have occurred during actual operation.

The fundamental problem of active, black-box learning of state diagrams (or automata) has been studied for decades. In 1956, Moore³¹ first

» key insights

- **Model learning aims to construct black-box state diagram models of software and hardware systems by providing inputs and observing outputs. The design of algorithms for model learning constitutes a fundamental research problem.**
- **Recently, much progress has been made in the design of new algorithms, both in a setting of finite state diagrams (Mealy machines) and in richer settings with data (register automata). Through the use of abstraction techniques, these algorithms can be applied to complex systems.**
- **Model learning is emerging as a highly effective bug-finding technique, with applications in areas such as banking cards, network protocols, and legacy software.**

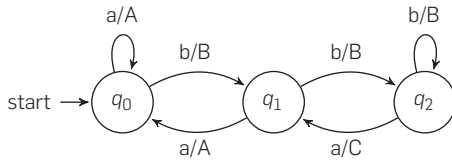


Mealy Machines

A (*deterministic*) Mealy machine is a tuple $\mathcal{M} = (I, O, Q, q_0, \delta, \lambda)$, where I is a finite set of inputs, O is a finite set of outputs, Q is a finite set of states, $q_0 \in Q$ is the initial state, $\delta: Q \times I \rightarrow Q$ is a transition function, and $\lambda: Q \times I \rightarrow O$ is an output function.

Figure 1 gives a graphical representation of a simple Mealy machine with inputs $\{a, b\}$, outputs $\{A, B, C\}$, states $\{q_0, q_1, q_2\}$, and initial state q_0 .

Figure 1. A simple Mealy machine.



Output function λ is extended to sequences of inputs by defining, for all $q \in Q$, $i \in I$, and $\sigma \in I^*$, $\lambda(q, \epsilon) = \epsilon$, and $\lambda(q, i\sigma) = \lambda(q, i)\lambda(\delta(q, i), \sigma)$. The behavior of Mealy machine \mathcal{M} is defined by function $A_{\mathcal{M}}: I^* \rightarrow O^*$ with $A_{\mathcal{M}}(\sigma) = \lambda(q_0, \sigma)$, for $\sigma \in I^*$. Mealy machines \mathcal{M} and \mathcal{N} are *equivalent*, denoted $\mathcal{M} \approx \mathcal{N}$, iff $A_{\mathcal{M}} = A_{\mathcal{N}}$. Sequence $\sigma \in I^*$ *distinguishes* \mathcal{M} and \mathcal{N} if and only if $A_{\mathcal{M}}(\sigma) \neq A_{\mathcal{N}}(\sigma)$.

proposed the problem of learning finite automata, provided an exponential algorithm and proved that the problem is inherently exponential. The problem has been studied under different names by different communities: control theorists refer to it as system identification, computation linguists speak about grammatical inference,²² some papers use the term regular inference,⁸ regular extrapolation,²⁰ or active automata learning,²⁴ and security researchers coined the term protocol state fuzzing.³⁴ Here, we will use the term *model learning* in analogy with the commonly used term model checking.¹⁵ Whereas model checking is widely used for analyzing finite-state models, model learning is a complementary technique for building such models from observed input–output data.

In 1987, Angluin⁶ published a seminal paper in which she showed that finite automata can be learned using the so-called *membership* and *equivalence queries*. Even though faster algorithms have been proposed since then, the most efficient learning algorithms that are being used today all follow Angluin’s approach of a *minimally adequate teacher (MAT)*. In the MAT framework, learning is viewed as a game in which a learner has to infer the behavior of an unknown state diagram

by asking queries to a teacher. The teacher knows the state diagram, which in our setting is a Mealy machine \mathcal{M} (see Mealy machines for the definition). Initially, the learner only knows the inputs I and outputs O of \mathcal{M} . The task of the learner is to learn \mathcal{M} through two types of queries:

- With a *membership query* (MQ), the learner asks what the output is in response to an input sequence $\sigma \in I^*$. The teacher answers with output sequence $A_{\mathcal{M}}(\sigma)$.
- With an *equivalence query* (EQ), the learner asks if a hypothesized Mealy machine \mathcal{H} with inputs I and outputs O is correct, that is, whether \mathcal{H} and \mathcal{M} are equivalent. The teacher answers *yes* if this is the case. Otherwise she answers *no* and supplies a *counterexample* $\sigma \in I^*$ that distinguishes \mathcal{H} and \mathcal{M} .

The L^* algorithm of Angluin⁶ is able to learn Mealy machine \mathcal{M} by asking a polynomial number of membership and equivalence queries (polynomial in the size of the corresponding canonical Mealy machine). In the Angluin’s algorithm, we give a simplified presentation of the L^* algorithm. Actual implementations, for instance in LearnLib²⁶ and libalf,⁹ contain many optimizations.

Peled et al.^{19,32} made the important observation that the MAT framework can be used to learn black box models of software and hardware components. Suppose we have a component, which we call the *System Under Learning (SUL)*, whose behavior can be described by (an unknown) Mealy machine \mathcal{M} . Suppose further that it is always possible to bring the SUL back to its initial state. A membership query can now be implemented by bringing the SUL to its initial state and then observing the outputs generated by the SUL in response to the given input sequence. Equivalence query can be approximated using a conformance testing (CT) tool²⁹ via a finite number of *test queries* (TQs). A test query asks for the response of the SUL to an input sequence, similar to a membership query. If one of the test queries exhibits a counterexample then the answer to the equivalence query is *no*, otherwise the answer is *yes*. A schematic overview is shown in Figure 4. In this approach, the task of the learner is to construct hypotheses, whereas the task of the conformance testing tool is to test the validity of these hypotheses. As a testing tool can only pose a finite number of queries, we can never be sure that a learned model is correct. However, a finite and complete conformance test suite does exist if we assume a bound on the number of states of machine \mathcal{M} .²⁹

The pioneering work of Peled et al.³² and Steffen et al.^{8,20,23} established fascinating connections between model learning and the area of formal methods, in particular model checking and model-based testing. Subsequent research has confirmed that, in the absence of a tractable white box model of a reactive system, a learned model is often an excellent alternative that may be obtained at relatively low cost.

In order to check properties of learned models, model checking¹⁵ can be used. In fact, Peled et al.³² showed how model learning and model checking can be fully integrated in an approach called *black box checking*. The basic idea is to use a model checker as a “preprocessor” for the conformance testing tool in Figure 4. When the teacher receives a hypothesis from the learner, it first runs a model checker to verify if the hypothesis model satisfies all the properties from the SUL’s specification. Only if this is true the

hypothesis is forwarded to the conformance tester. If one of the properties does not hold then the model checker produces a counterexample. Now there are two cases. The first possibility is that the counterexample can be reproduced on the SUL. This means we have demonstrated a bug in the SUL (or in its specification) and we stop learning. The second possibility is that the counterexample cannot be reproduced on the SUL. In this case the teacher returns the counterexample to the learner since it follows that the hypothesis is incorrect. In later work,^{16,19} the black box checking approach has been further refined and it has been successfully applied to several industrial cases.

The required number of membership queries of most learning algorithms grows linearly with the number of inputs and quadratically with the number of states.²⁴ This means that learning algorithms scale rather well when the number of inputs grows; in other words, formulating a new hypothesis is easy. However, checking that a hypothesis is correct (conformance testing), quickly becomes a bottleneck for larger numbers of inputs. If the current hypothesis has n states, the SUL has n' states, and there are k inputs, then in the worst case we need to run test sequences that contain all possible sequences of n' – n inputs, that is, $k^{(n' - n)}$ possibilities.²⁹ As a result, model learning currently can only be applied if there are less than, say, 100 inputs. Thus, we seek methods that help us to reduce the number of inputs.

Abstraction is the key for scaling model learning methods to realistic applications. Cho et al.¹⁴ succeeded to infer models of realistic botnet command and control protocols by placing an emulator/mapper between botnet servers and the learning software, which concretizes the alphabet symbols into valid network messages and sends them to botnet servers. When responses are received, the emulator does the opposite—it abstracts the response messages into the output alphabet and passes them on to the learning software. A schematic overview of this learning setup is shown in Figure 5. The idea of an intermediate mapper component that takes care of abstraction is very natural and is used, implicitly or explicitly, in many case studies on automata learning. Aarts

Angluin’s Algorithm

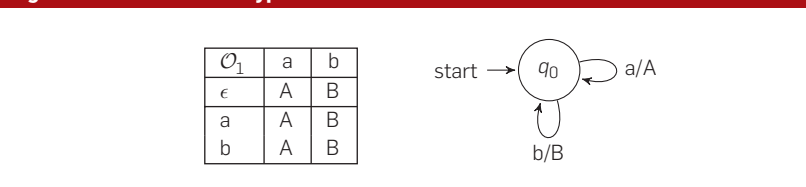
The L^* algorithm incrementally constructs an observation table with entries taken from the set O of outputs. The rows are labeled by words in $S \cup S \cdot I$, where S is a nonempty finite prefix-closed language, and the columns by a nonempty finite suffix-closed language E . Formally, an observation table is a triple (S, E, row) , where $row: S \cup (S \cdot I) \rightarrow (E \rightarrow O)$. For a given prefix w and suffix e , $row(w)(e)$ returns the last output produced by the SUL in response to the membership query we . Initially, S only contains the empty word ϵ , and E equals set of inputs I .

Two crucial properties of the observation table allow for the construction of a Mealy machine: closedness and consistency. Observation table (S, E, row) is *closed* if for all $w \in S \cdot I$ there is a $w' \in S$ with $row(w) = row(w')$. It is *consistent* if whenever $row(w_1) = row(w_2)$ for some $w_1, w_2 \in S$, then $row(w_1 a) = row(w_2 a)$ for all $a \in I$.

If a table is closed and consistent, the learner constructs a Mealy machine $\mathcal{H} = (I, O, Q, q_0, \delta, \lambda)$ with $Q = \{row(w) \mid w \in S\}$, $q_0 = row(\epsilon)$, $\delta(row(w), a) = row(w \cdot a)$, and $\lambda(row(w), a) = row(w)(a)$.

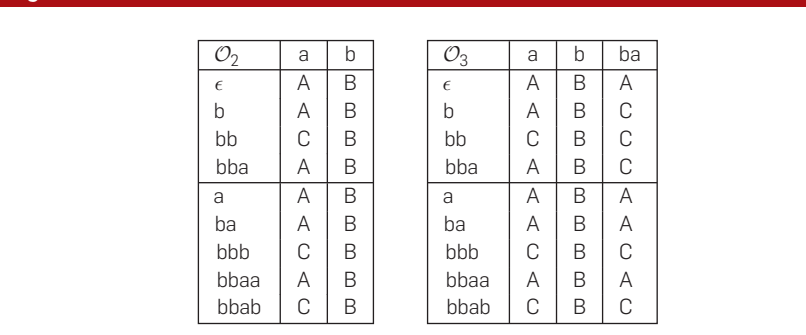
Assume the teacher knows the Mealy machine \mathcal{M} from Figure 1. The learner starts to ask queries to fill the initial table. The result is shown in Figure 2 (left). As this table is both closed and consistent, the learner constructs an initial hypothesis \mathcal{H} , shown in Figure 2 (right).

Figure 2. First table and hypothesis \mathcal{H} .



Hypothesis \mathcal{H} is incorrect since, for instance, sequence bba distinguishes \mathcal{H} from \mathcal{M} . Assume that the teacher returns counterexample bba to the learner. To process this counterexample, the learner adds bba and all its prefixes to S and constructs the table shown in Figure 3 (left). Since $row(\epsilon) = row(b)$ but $row(b)(a) \neq row(bb)(a)$ this table is not consistent. Thus, we add ba to set E and obtain the table shown in Figure 3 (right). This table is closed and consistent, and the corresponding Mealy machine is equivalent to \mathcal{M} .

Figure 3. Second and third table.



et al.² developed a mathematical theory of such intermediate abstractions, with links to predicate abstraction and abstract interpretation.

A complementary, simple but

practical approach is to apply model learning for multiple smaller subsets of inputs. This will significantly reduce the learning complexity, also because the set of reachable states will typically

be smaller for a restricted number of stimuli. Models learned for a subset of the inputs may then be used to generate counterexamples while learning models for larger subsets. Yet another approach, which, for instance, has been applied by Chalupar et al.,¹³ is to merge several input actions that usually occur in a specific order into a single

high-level action, thus reducing the number of inputs. Again, models that have been learned with a small number of high level inputs may be used to generate counterexamples in subsequent experiments in which these inputs are broken up into their constituents.

Paraphrasing C.A.R. Hoare, one could say that in every large program there is

a small state machine trying to get out. By choosing a proper set of input actions and by defining an appropriate mapper/abstraction, we can make this small state machine visible to the learner.

Examples of Applications

During recent years, model learning has been successfully applied to numerous practical cases in different domains. There have been industrial applications, for instance, on regression testing of telecommunication systems at Siemens,²⁰ on integration testing at France Telecom,³⁶ on automatic testing of an online conference service of Springer Verlag,³⁹ and on testing requirements of a brake-by-wire system from Volvo Technology.¹⁶ Below, I review some representative case studies that have been carried out at Radboud University related to smart cards, network protocols, and legacy software.

Smartcards. Chalupar et al.¹³ used model learning to reverse engineer the e.dentifier2, a smartcard reader for Internet banking. To be able to learn a model of the e.dentifier2, the authors constructed a Lego robot, controlled by a Raspberry Pi that can operate the keyboard of the reader (see Figure 6). Controlling all this from a laptop, they then could use LearnLib²⁶ to learn models of the e.dentifier2. They learned a four-state Mealy machine of one version of the e.dentifier2 that revealed the presence of a security flaw, and showed that the flaw is no longer present in a three-state model for the new version of the device.

In another study, Aarts et al.³ learned models of implementations of the EMV protocol suite on bank cards issued by several Dutch and German banks, on MasterCard credit cards issued by Dutch and Swedish banks, and on one UK Visa debit card. To learn the models, LearnLib performed between 855 and 1,696 membership and test queries for each card and produced models with four to eight states. (Figure 7 shows one of the learned models.) All cards resulted in different models, only the applications on the Dutch cards were identical. The models learned did not reveal any security issues, although some peculiarities were noted. The authors argue that model learning would be useful as part of security evaluations.

Network protocols. Our society has become completely dependent on the

Figure 4. Model learning within the MAT framework.

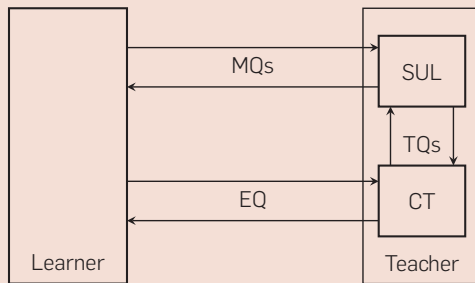


Figure 5. Model learning with a mapper.

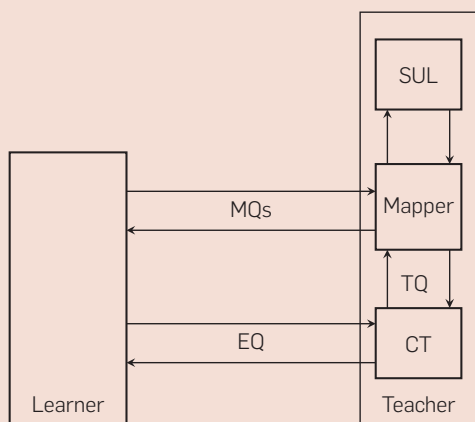
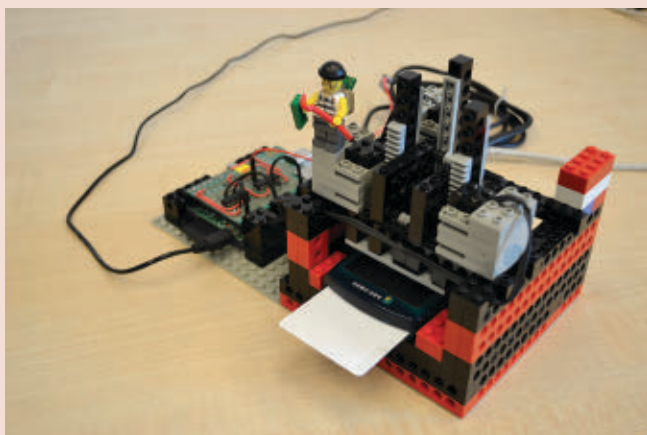


Figure 6. Lego robot used to reverse engineer the e.dentifier2 smartcard reader (picture courtesy of Chalupar¹³).



correct functioning of network and security protocols. Bugs or vulnerabilities in these protocols may lead to security breaches or even complete network failures. Model checking¹⁵ has proven to be an effective technique for finding such bugs and vulnerabilities. However, since exhaustive model checking of protocol implementations is usually not feasible,²⁷ model checking is usually applied to models that have been handcrafted starting from protocol standards. This means that model checking is unable to catch bugs that arise because implementations do not conform to their specification. Model learning turns out to be effective in finding exactly this type of bugs, which makes the technique complementary to model checking.

De Ruiter and Poll,³⁴ for instance, analyzed both server- and client-side implementations of the TLS protocol with a test harness that supported several key exchange algorithms and the option of client certificate authentication. They showed that model learning (or protocol state fuzzing, as they call it) can catch an interesting class of implementation flaws that is apparently common in security protocol implementations: in three out of nine tested TLS implementations new security flaws were found. For the Java Secure Socket Extension, for instance, a model was learned for Java version 1.8.0.25. The authors observed that the model contained *two* paths leading to the exchange of application data: the regular TLS protocol run and another unexpected run. By exploiting this behavior, an attack was possible in which both the client and the server application would think they were talking on a secure connection, where in reality anyone on the line could read the client's data and tamper with it. A fix was released as part of a critical security update, and by learning a model of JSSE version 1.8.0.31, the authors were able to confirm that indeed the problem was solved. Due to a manually constructed abstraction/mapper, the learned Mealy machines were all quite small, with 6–16 states. As the analysis of different TLS implementations resulted in different and unique Mealy machines for each one, model learning could also be used for fingerprinting TLS implementations.

Fiterău et al.¹⁷ combined model learning and model checking in a case study involving Linux, Windows, and FreeBSD implementations of TCP servers and clients. Model learning was used to infer models of different components and then model checking was applied to fully explore what may happen when

these components (for example a Linux client and a Windows server) interact. The case study revealed several instances in which TCP implementations do not conform to their RFC specification, see Figure 8 for an example.

Legacy software. Legacy systems have been defined as “large software systems

Figure 7. State machine of SecureCode Aut application on Dutch Rabo bank card (diagram courtesy of Aarts et al.³).

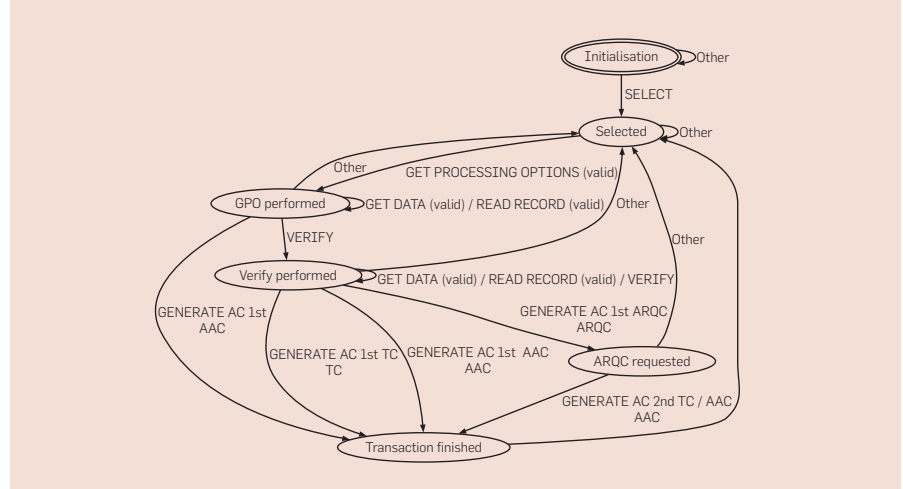
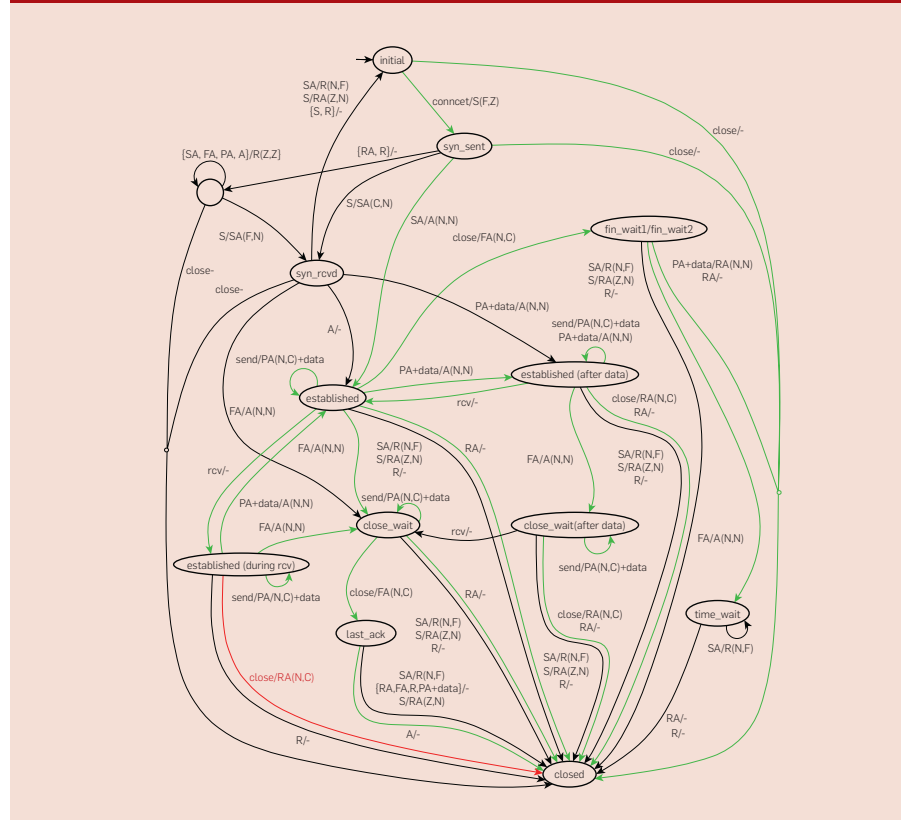


Figure 8. Learned state machine for Windows8 TCP Client (picture courtesy of Grace et al.¹⁹). Transitions that are reachable when the Windows8 client interacts with a Windows8 server in a setting with reliable communication are colored green (as computed by a model checker). The red transition marks a nonconformance to the RFC: a Close can generate a RST instead of a FIN even in cases where there is no data to be received, namely, in states where a rcv call is pending.



that we do not know how to cope with but that are vital to our organization.”⁷ Typically, these systems are based on obsolete technologies, documentation is limited, and the original developers are no longer available. In addition, existing regression tests will be limited. Given these characteristics, innovations that require changes of legacy components are risky. Several techniques have been developed to extract the crucial business information hidden in legacy components, and to support the construction of refactored implementations. Margaria et al.³⁰ were the first to point out that model learning may help to increase confidence that a legacy component and a refactored implementation have the same behavior.

Schuts et al.,³⁵ for instance, used model learning to support the rejuvenation of legacy embedded software in a development project at Philips. The project concerned the introduction of a new hardware component, the Power Control Component (PCC), which is used to start-up and shutdown an interventional radiology system. All computers in the system

have a software component, the Power Control Service (PCS) which communicates with the PCC over an internal control network during the execution of start-up and shutdown scenarios. To deal with the new hardware of the PCC, which has a different interface, a new implementation of the PCS was needed. Since different configurations had to be supported, with old and new PCC hardware, the old and new PCS software needed to have exactly the same external behavior. Figure 9 illustrates the approach that was followed. From both the legacy implementation A and the refactored implementation B , Mealy machine models \mathcal{M}_A resp. \mathcal{M}_B were obtained using model learning. These models were then compared using an equivalence checker. When the equivalence checker found a counterexample σ , then we checked whether A and \mathcal{M}_A behaved the same on input σ and whether B and \mathcal{M}_B behaved the same on input σ . If there was a discrepancy between A and \mathcal{M}_A , or between B and \mathcal{M}_B , then we asked the learner to construct an improved model based on counterexample σ . Otherwise σ

exhibited a difference between A and B , and we changed either A or B (or both), depending on which response to σ was considered unsatisfactory behavior. The implementations were learned and checked iteratively with increasing sets of stimuli to handle scalability. Issues were found in both the refactored and the legacy implementation in an early stage, before the component was integrated. In this way, costly rework in a later phase of the development was avoided.

Recent Advances

During recent years significant progress has been made on algorithms for model learning, which is crucial for scaling the application of these techniques to larger systems.

Basic algorithms. Since 1987, the L^* algorithm of Angluin’s⁶ has been considerably improved. The original L^* performs a membership query for each entry in the observation table. This is often redundant, given that the sole purpose of membership queries is the distinction of states (rows). Therefore, Kearns and Vazirani²⁸ replaced the observation table of the L^* algorithm by the so-called discrimination trees, which are basically decision trees for determining equivalence of states.

Another inefficiency of L^* is that all prefixes of a counterexample are added as rows to the table. Counterexamples obtained through conformance testing or runtime monitoring may be extremely long and are rarely minimal, which results in numerous redundant membership queries. Rivest and Schapire³³ observed that, instead of adding all prefixes of a counterexample as rows to the table, it suffices to add a single, well-chosen suffix as a column.

The new TTT algorithm of Isberner et al.^{24, 25} is currently the most efficient algorithm for active learning. The algorithm builds on the ideas of Kearns and Vazirani²⁸ and Rivest and Schapire³³ but eliminates overly long discrimination trees, which may arise when processing long counterexamples, by cleaning up the internal data structures and reorganizing the discrimination tree. Suppose that a Mealy machine \mathcal{M} has n states and k inputs, and that the length of the longest counterexample returned by the teacher is m . Then in the worst-case TTT requires $O(n)$ equivalence queries and $O(kn^2 +$

Figure 9. Approach to compare legacy component and refactored implementation (diagram courtesy of Schuts et al.³⁵).

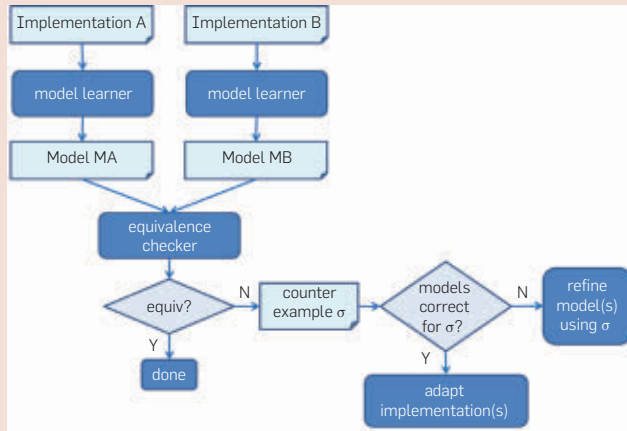
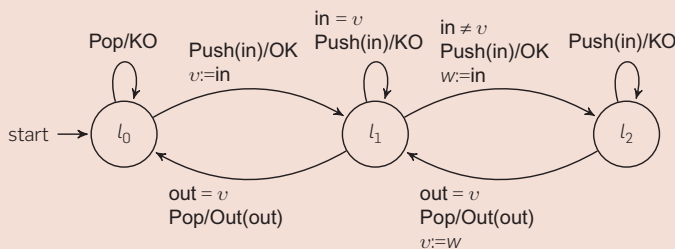


Figure 10. A register automaton.



$n \log m$) membership queries, each of length $O(n + m)$. This worst-case query and symbol complexity coincides with the algorithm of Rivest and Schapire,³³ but TTT is faster in practice.

The TTT algorithm typically generates more intermediate hypotheses than the L^* algorithm. This suggests that the number of input symbols used in membership queries alone may not be an appropriate metric for comparing learning algorithms: we also need to take into account the number of test queries required to implement equivalence queries. The total number of input symbols in membership and test queries appears to be a sensible metric to compare learning approaches in practice. Two of my students, J. Moerman and A. Fedotov, compared different combinations of learning and testing algorithms on a large number of benchmarks (protocols, control software, circuits, etc.) and found that TTT used on average 3.9 times fewer input symbols than L^* .

Learning and testing can be easily parallelized when it is possible to run multiple instances of the SUL concurrently. Another technique that may speedup learning is to save and restore software states of the SUL (checkpointing). The benefit is that when the learner wants to explore different outgoing transitions from a saved state q it only needs to restore q , which usually is much faster than resetting the system and bringing it back to q via a sequence of inputs. Henrix²¹ reports on experiments in which checkpointing with DMTCP speeds up learning with a factor 1.7.

Register automata. Even though we have seen much progress on basic algorithms for learning state machines, these algorithms only succeed to learn relatively small state machines. In order to scale the application of these algorithms to realistic applications, users typically need to manually construct abstractions or mappers.² This can be a time-consuming activity that requires several iterations and expert knowledge of the SUL. Therefore, much work has been carried out recently to generalize learning algorithms to richer classes of models that have more structure, in particular EFSM models in which data values may be communicated, stored, and manipulated.

One particular extension for which model learning algorithms have been

Abstraction is the key for scaling model learning methods to realistic applications.

developed is that of *register automata*.¹¹ These automata have a finite set of states but are extended with a set of registers that can be used to store data values. Input and output actions are parameterized by data values, which may be tested for equality in transition guards and stored in registers. Figure 10 gives a simple example of a register automaton, a FIFO-set with capacity two. A FIFO-set corresponds to a queue in which only different values can be stored. There is a $\text{Push}(d)$ input symbol that tries to insert a value d in the queue, and a Pop input symbol that tries to retrieve a value from the queue. The output in response to a Push is OK if the input value can be added successfully, or KO if the input value is already in the queue or if the queue is full. The output in response to a Pop is Out, with as parameter the oldest value from the queue, or KO if the queue is empty.

In register automata all data values are fully symmetric, and this symmetry may be exploited during learning. Two different approaches have been explored in the literature. A first approach, followed by Cassel et al.,¹² has been implemented in the software tools LearnLib²⁶ and RALib.¹⁰ Model learning algorithms usually rely on the Nerode relation for identifying the states and relations of a learned automaton: two words lead to the same state if their residual languages coincide. The basic idea now is to formulate a Nerode-like congruence for register automata, which determines the states, transitions, and registers of the inferred automaton. Technical basis of the implementation are the so-called symbolic decision trees, which can be used to summarize the results of many tests using a concise symbolic representation.

A second approach for learning register automata, followed by Aarts et al.¹ has been implemented in the software tool Tomte. In this approach, counterexample-guided abstraction refinement is used to automatically construct an appropriate mapper. The idea is to start with a drastic abstraction that completely ignores the data values that occur in input and output actions. When this abstraction is too coarse, the learner will observe non-deterministic behavior. In the example of Figure 10, for instance, an input


sequence Push Push Pop Pop will mostly trigger outputs OK OK Out KO, but sometimes OK OK Out Out. Analysis of this behavior will then lead to a refinement of the abstraction. In our example, for instance, we need at least two abstract versions of the second Push, since apparently it matters whether or not the data value of this input is equal to the data value of the first Push. RALib and Tomte both outperform LearnLib. The performance of Tomte and RALib is roughly comparable. RALib outperforms Tomte on some benchmarks, but Tomte is able to learn some register automata that RALib cannot handle, such as a FIFO-set with capacity 40.

Research Challenges


Even though model learning has been applied successfully in several domains, the field is still in its infancy. There is a huge potential for applications, especially in the area of legacy control software, but more research on algorithms and tools is needed to bring model learning from the current level of academic prototypes to that of an off-the-shelf technology that can be easily applied to a large class of systems. Here, I discuss some of the major research challenges.

Predicates and operations on data.

The recent extension of model learning algorithms to register automata is a breakthrough which, potentially, makes model learning applicable to a much larger class of systems. Due to the restriction that no operations on data are allowed, the class of systems that can be described as register automata is small, and mainly consists of academic examples such as the bounded retransmission protocol and some simple data structures. However, as pointed out by Cassel et al.,¹² using SMT solving the new learning algorithms for register automata can be extended to EFSM formalisms in which guards may contain predicates such as the successor and less than relation. A prototype implementation RALib is available and we are close to the point where we can learn models of real-world protocols such as TCP, SIP, SSH, and TLS automatically, without the need to manually define abstractions. Nevertheless, our understanding of algorithms for learning EFSMs with different predicates and



Even though model learning has been applied successfully in several domains, the field is still in its infancy.



operations is still limited, and there are many open questions.

Isberner²⁴ developed a model learning algorithm for *visibly pushdown automata (VPAs)*, a restricted class of pushdown automata proposed by Alur and Madhusudan.⁵ This result is in a sense orthogonal to the results on learning register automata: using register automata learning, a stack with a finite capacity storing values from an infinite domain can be learned, whereas using VPA learning it is possible to learn a stack with unbounded capacity storing data values from a finite domain. From a practical perspective it would be useful to develop a learning algorithm for a class of models that generalizes both register automata and VPAs. There are many protocols in which messages may be buffered, and we therefore need algorithms that can learn queues with unbounded capacity.

Beyond Mealy machines. In a Mealy machine, a single input always triggers a single output. In practice, however, a system may respond to an input with zero or more outputs. Moreover, the behavior of systems is often timing dependent and a certain output may only occur if some input has not been offered for a certain amount of time. As a consequence, practical application of model learning is often severely restricted by the lack of expressivity of Mealy machines. For instance, in order to squeeze TCP implementations into a Mealy machine, we had to eliminate timing-based behavior as well as retransmissions.¹⁷ There has been some preliminary work on extending learning algorithms to I/O automata⁴ and to event-recording automata,¹⁸ but a major effort is still required to turn these ideas into practical tools.

Systems are often nondeterministic, in the sense that a sequence of inputs may lead to different output events in different runs. Existing model learning tools, however, are only able to learn deterministic Mealy machines. In applications, we can sometimes eliminate nondeterminism by abstracting different concrete output events into a single abstract output, but in many cases this is not possible. Volpato and Tretmans³⁸ present an adaptation of L^* for active learning of nondeterministic I/O automata. Their algorithm enables learning of nondeterministic SULs, and it allows us to construct partial or approximate

models. Again, a major effort will be required to incorporate these ideas in state-of-the-art tools such as LearnLib, libalf, RALib, or Tomte.

Quality of models. Since the models produced by model learning algorithms have been obtained through a finite number of tests, we can never be sure that they are correct. Nevertheless, from a practical perspective, we would like to be able to make quantitative statements about the quality of learned models and, for instance, assert that a hypothesis is approximately correct with high probability. Angluin⁶ proposed such a setting, along the lines of the PAC learning approach of Valiant.³⁷ Her idea was to assume some (unknown) probability distribution on the set of words over the input alphabet I . In order to test a hypothesis, the conformance tester (see Figure 4) selects a specified number of input words (these are statistically independent events) and checks for each word whether the resulting output of SUL and hypothesis agrees. Only when there is full agreement the conformance tester returns answer *yes* to the learner. An hypothesis is said to be an ϵ -approximation of the SUL if the probability of selecting a string that exhibits a difference is at most ϵ . Given a bound on the number of states of the SUL, and two constants ϵ and δ , Angluin's polynomial algorithm produces a model such that the probability that this model is an ϵ -approximation of the SUL is at least $1 - \delta$. Angluin's result is elegant but not realistic in a setting of reactive systems, since there we typically do not have a fixed distribution over the input words. (Inputs are under the control of the environment of the SUL, and this environment may change.)

Using traditional conformance testing,²⁹ we can devise a test suite that can guarantee the correctness of a learned model, given an upper bound on the number of states of the SUL. But such an approach is also not satisfactory, since the required number of test sequences grows exponentially with the number of states of the SUL. The challenge therefore is to establish a middle ground between Angluin's approach and traditional conformance testing. Systems logs often provide a probability distribution on the set of

input words that may be used as a starting point for defining a metric.

Opening the box. There can be many reasons for using black box model learning techniques. For instance, we may want to understand the behavior of a component but do not have access to the code. Or we may have access to the code but not to adequate tools for analyzing it (for example, in the case of legacy software). Even in "white box" situations where we have access both to the code and to powerful code analysis tools, black box learning can make sense, for instance because a black box model can be used to generate regression tests, for checking conformance to a standard, or as part of model-based development of a larger system. An important research challenge is to combine black box and white box model extraction techniques and, for instance, to use white box methods such as static analysis and concolic testing to help answering equivalence queries posed by a black box learner.

Acknowledgments. Portions of this work were performed in the context of STW projects 11763 (ITALIA) and 13859 (SUMBAT), and NWO projects 628.001.009 (LEMMA) and 612.001.216 (ALSEP). ■

References

- Aarts, F., Fiterău-Broștean, P., Kuppens, H., Vaandrager, F. Learning register automata with fresh value generation. In *ICTAC'15*, LNCS 9399 (2015). Springer, 165–183.
- Aarts, F., Jonsson, B., Uijen, J., Vaandrager, F. Generating models of infinite-state communication protocols using regular inference with abstraction. *Formal Methods Syst. Des.* 46, 1 (2015), 1–41.
- Aarts, F., de Ruiter, J., Poll, E. Formal models of bank cards for free. In *SECTEST'13* (2013). IEEE, 461–468.
- Aarts, F., Vaandrager, F. Learning I/O automata. In *CONCUR'10*, LNCS 6269 (2010). Springer, 71–85.
- Alur, R., Madhusudan, P. Visibly pushdown languages. In *STOC'04* (2004). ACM, 202–211.
- Angluin, D. Learning regular sets from queries and counterexamples. *Inf. Comput.* 75, 2 (1987), 87–106.
- Bennett, K. Legacy systems: coping with success. *IEEE Softw.* 12, 1 (1995), 19–23.
- Berg, T., Grinchtein, O., Jonsson, B., Leucker, M., Raffelt, H., Steffen, B. On the correspondence between conformance testing and regular inference. In *FASE'05*, LNCS 3442 (2005). Springer, 175–189.
- Bollig, B., Katoen, J.-P., Kern, C., Leucker, M., Neider, D., Piegdon, D. libalf: The automata learning framework. In *CAV'10*, LNCS 6174 (2010). Springer, 360–364.
- Cassel, S., Howar, F., Jonsson, B. RALib: A LearnLib extension for inferring EFSMs. In *DIFTS'15* (2015).
- Cassel, S., Howar, F., Jonsson, B., Merten, M., Steffen, B. A succinct canonical register automaton model. *J. Log. Algebr. Meth. Program.* 84, 1 (2015), 54–66.
- Cassel, S., Howar, F., Jonsson, B., Steffen, B. Active learning for extended finite state machines. *Formal Asp. Comput.* 28, 2 (2016), 233–263.
- Chalupar, G., Peherstorfer, S., Poll, E., Ruiter, J. Automated reverse engineering using Lego. In *WOOT'14* (Aug. 2014). IEEE Computer Society.
- Cho, C., Babic, D., Shin, E., Song, D. Inference and analysis of formal models of botnet command and control protocols. In *CCS'10* (2010). ACM, 426–439.
- Clarke, E., Grumberg, O., Peled, D. *Model Checking*. MIT Press, Cambridge, MA, 1999.
- Feng, L., Lundmark, S., Meinke, K., Niu, F., Sindhu, M., Wong, P. Case studies in learning-based testing. In *ICTSS'13*, LNCS 8254 (2013). Springer, 164–179.
- Fiterău-Broștean, P., Janssen, R., Vaandrager, F. Combining model learning and model checking to analyze TCP implementations. In *CAV'16*, LNCS 9780 (2016). Springer, 454–471.
- Grinchtein, O., Jonsson, B., Leucker, M. Learning of event-recording automata. *Theor. Comput. Sci.* 411, 47 (2010), 4029–4054.
- Groce, A., Peled, D., Yannakakis, M. Adaptive model checking. *Logic J. IGPL* 14, 5 (2006), 729–744.
- Hagerer, A., Hungar, H., Niese, O., Steffen, B. Model generation by moderated regular extrapolation. In *FASE'02*, LNCS 2306 (2002). Springer, 80–95.
- Henrix, M. Performance improvement in automata learning. Master thesis, Radboud University (2015).
- Hungar, H., Niese, O., Steffen, B. Domain-specific optimization in automata learning. In *CAV'03*, LNCS 2725 (2003). Springer, 315–327.
- de la Higuera, C. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.
- Isberner, M. Foundations of active automata learning: An algorithmic perspective. PhD thesis, Technical University of Dortmund (2015).
- Isberner, M., Howar, F., Steffen, B. The TTT algorithm: A redundancy-free approach to active automata learning. In *RV'14*, LNCS 8734 (2014). Springer, 307–322.
- Isberner, M., Howar, F., Steffen, B. The open-source LearnLib – A framework for active automata learning. In *CAV'15*, LNCS 9206 (2015). Springer, 487–495.
- Jhala, R., Majumdar, R. Software model checking. *ACM Comput. Surv.* 41, 4 (Oct. 2009), 21:1–21:54.
- Kearns, M.J., Vazirani, U.V. *An Introduction to Computational Learning Theory*. MIT Press, 1994.
- Lee, D., Yannakakis, M. Principles and methods of testing finite state machines—A survey. *Proc. IEEE* 84, 8 (1996), 1090–1123.
- Margaria, T., Niese, O., Raffelt, H., Steffen, B. Efficient test-based model generation for legacy reactive systems. In *HLDVT'04* (2004). IEEE Computer Society, 95–100.
- Moore, E. Gedanken-experiments on sequential machines. In *Automata Studies*, Annals of Mathematics Studies 34 (1956). Princeton University Press, 129–153.
- Peled, D., Vardi, M., Yannakakis, M. Black box checking. *J. Autom. Lang. Comb.* 7, 2 (2002), 225–246.
- Rivest, R.L., Schapire, R.E. Inference of finite automata using homing sequences. *Inf. Comput.* 103, 2 (1993), 299–347.
- de Ruiter, J., Poll, E. Protocol state fuzzing of TLS implementations. In *USENIX Security'15* (2015). USENIX Association, 193–206.
- Schuts, M., Hooman, J., Vaandrager, F. Refactoring of legacy software using model learning and equivalence checking: an industrial experience report. In *IFM'16*, LNCS 9681 (2016). Springer, 311–325.
- Shahbaz, M., Groz, R. Analysis and testing of black-box component-based systems by inferring partial models. *Softw. Test. Verif. Reliab.* 24, 4 (2014), 253–288.
- Valiant, L.G. A theory of the learnable. In *STOC'84* (1984). ACM, 436–445.
- Volpatò, M., Tretmans, J. Approximate active learning of nondeterministic input output transition systems. *Electron. Commun. EASST* 72 (2015).
- Windmüller, S., Neubauer, J., Steffen, B., Howar, F., Bauer, O. Active continuous quality control. In *CBSE'13* (2013). ACM, 111–120.

Frits Vaandrager (F.Vaandrager@cs.ru.nl), Department of Software Science, Institute for Computing and Information Sciences at Radboud University, Nijmegen, The Netherlands.

© 2017 ACM 0001-0782/17/02 \$15.00



Watch the author discuss his work in this exclusive *Communications* video. <http://cacm.acm.org/videos/model-learning>

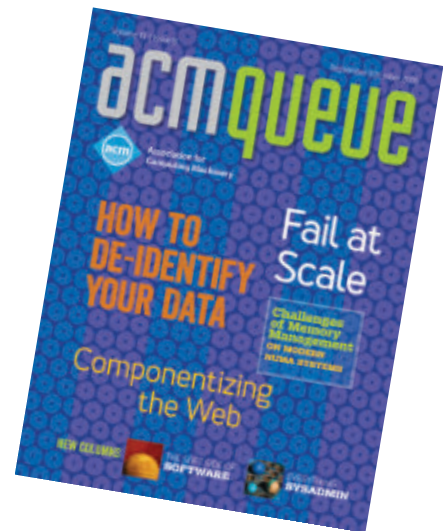
acmqueue

Check out the new acmqueue app

FREE TO ACM MEMBERS

acmqueue is ACM's magazine by and for practitioners, bridging the gap between academics and practitioners of computer science. After more than a decade of providing unique perspectives on how current and emerging technologies are being applied in the field, the new acmqueue 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.

P. 98

**Technical
Perspective
Cleaning Up
Flaws in TLS
Implementations**

By Eric Rescorla

P. 99

A Messy State of the Union: Taming the Composite State Machines of TLS

By Benjamin Beurdouche, Karthikeyan Bhargavan,
Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss,
Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue

P. 108

Authentication Using Pulse-Response Biometrics

By Ivan Martinovic, Kasper Rasmussen,
Marc Roeschlin, and Gene Tsudik

Technical Perspective

Cleaning Up Flaws in TLS Implementations

By Eric Rescorla

ONE OF THE unfortunate facts about protocols is that as they get older and applied to more application scenarios—and TLS is used basically everywhere—they tend to gain weight. SSLv3 was not small to begin with: When it was designed it supported static RSA, ephemeral RSA, static Diffie-Hellman, ephemeral Diffie-Hellman, Fortezza, as well as a session resumption mode. Over the past 20 years or so, the IETF also added Elliptic Curve cipher suites, Kerberos, SRP, and 25 or so “extension” code points, ranging from Server Name Indication to token binding.

It is a truism in the security community that “complexity is the enemy of security” and the sheer surface area of TLS has historically made people uncomfortable, but what the miTLS team has shown is that all this stuff represents a real threat to user security. At a general level this is not surprising; as Steven Bellovin has said, “software has bugs and security software has security relevant bugs.” What’s new here? Two things: First, a general methodology for finding this kind of defect and a demonstration that it can find them on real systems. Second, the miTLS team has shown that having a large set of modes of various strengths was dangerous, even if your software is configured to favor the strong modes of operation


or to only have strong modes—this was surprising (and bad) news to everyone who thought that they had protected themselves by disabling those weak modes!

What is the community doing about this problem?

First, implementors are moving rapidly to downsize their TLS stacks. OpenSSL recently shipped version 1.1.0, containing a major revision of their state machine. Two OpenSSL forks, BoringSSL and LibreSSL have embarked upon more radical surgery, removing such features as Kerberos and SSLv2 (and in the case of LibreSSL, SSLv3). Similarly, NSS, the

It’s a truism in the security community that “complexity is the enemy of security” and the sheer surface area of TLS has historically made people uncomfortable.

stack used by Firefox, has removed support for SSLv2 and disabled SSLv3 by default. Browser vendors have been even more aggressive: all four major browsers now disable RC4; IE, Chrome and Firefox no longer support SSLv3; and Chrome recently disabled finite field Diffie-Hellman, leaving only Elliptic Curve cipher suites for sites that want forward secrecy.

Second, we are in the process of simplifying TLS itself. The IETF is nearing completion of TLS 1.3, the first really major revision of TLS since the development of SSLv3. TLS 1.3 started by drastically reducing the number of cryptographic variants: converging on Diffie-Hellman key exchange (in a small number of predefined Elliptic Curve and Finite Field groups), authenticated by digital signatures or a pre-shared key. Removed features include: static RSA, static Diffie-Hellman, compression, stream ciphers, CBC-mode block ciphers, SRP, Kerberos, and everyone’s favorite, renegotiation. The result is a version of TLS that is hopefully both stronger—because we have removed many dangerous and problematic features—and easier to implement and analyze. 

Eric Rescorla (ekr@rtfm.com) works on Firefox at Mozilla.

Copyright held by author.

A Messy State of the Union: Taming the Composite State Machines of TLS

By Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue

Abstract

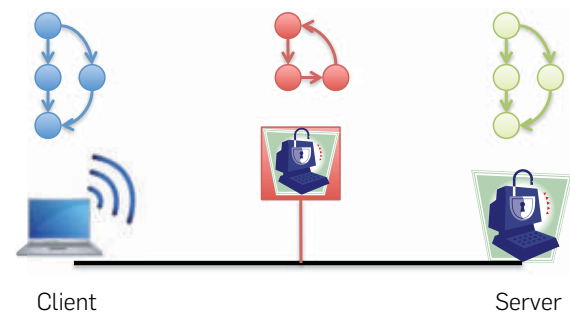
The Transport Layer Security (TLS) protocol supports various authentication modes, key exchange methods, and protocol extensions. Confusingly, each combination may prescribe a different message sequence between the client and the server, and thus a key challenge for TLS implementations is to define a composite state machine that correctly handles these combinations. If the state machine is too restrictive, the implementation may fail to interoperate with others; if it is too liberal, it may allow unexpected message sequences that break the security of the protocol. We systematically test popular TLS implementations and find unexpected transitions in many of their state machines that have stayed hidden for years. We show how some of these flaws lead to critical security vulnerabilities, such as FREAK. While testing can help find such bugs, formal verification can prevent them entirely. To this end, we implement and formally verify a new composite state machine for OpenSSL, a popular TLS library.

1. TRANSPORT LAYER SECURITY

Transport Layer Security (TLS),¹³ previously known as Secure Sockets Layer (SSL), is a standard cryptographic protocol widely used to secure communications for the web (HTTPS), email, and wireless networks. Figure 1 depicts the common usage of TLS and its threat model. Following the protocol, a client and a server exchange messages to establish a secure channel across an insecure network. Meanwhile, a network attacker can intercept these messages, tamper with them, and inject new messages to confuse the two. Additionally, the attacker may control some malicious clients and servers that are free to deviate from the protocol. The goal of TLS is to ensure the integrity and confidentiality of data exchanged between honest clients and servers, despite the best efforts of attackers.

TLS offers a large choice of cryptographic algorithms and protocol features to accommodate the needs of diverse applications. Each TLS connection consists of a channel establishment protocol, called the *handshake*, followed by a transport protocol, the *record*. During the handshake, the client and server negotiate which algorithms and features they wish to use. For example, the client and server may be authenticated with certificates, or with pre-shared keys, or may remain anonymous; the key exchange may use Ephemeral Diffie-Hellman or RSA Encryption; the record protocol may encrypt sensitive application data using AES-GCM or RC4.

Figure 1. TLS threat model: network attacker aims to subvert client-server exchange.



If a connection uses a secure key exchange and a strong record encryption scheme, security against network attackers can be reduced to the security of these building blocks. Indeed, recent works provide cryptographic proofs for some of the key exchange methods^{7, 16, 19, 22} and encryption schemes²⁷ used in TLS. However, not all of choices offered by TLS have been proved secure; in fact, many of them are obsolete and some are even known to be broken. Still, TLS client and servers continue to support old protocol versions, extensions, and ciphersuites for interoperability reasons. For example, TLS 1.0 offered several deliberately weakened ciphersuites to comply with US export regulations at the time. These ciphersuites were explicitly deprecated in TLS 1.1 but are still supported by mainstream implementations.

Even if the client and server support weak protocol modes, the TLS handshake is designed to negotiate and execute the strongest protocol that they both support. Hence, if *one* party is configured to accept only strong parameters, then its connections are expected to be secure, even if its peer supports other weaker modes. However, this guarantee depends on the implementation correctly composing different protocol modes, a task that is surprisingly tricky.

1.1. Composing protocol state machines

Each TLS client and server implements a state machine that keeps track of the protocol being run: which messages

The original version of this paper was published in *IEEE Symposium on Security and Privacy*, 2015, pages 535–552.

have been sent and received, what cryptographic materials have been computed, which messages are expected next, etc. The state machine for each individual ciphersuite is specified in the standard, but the task of writing a composite state machine for multiple ciphersuites is left to each implementation.

Figure 2 depicts a simplified TLS handshake for some (fictional) ciphersuite, as seen from the viewpoint of the client. On the left, the client first sends a `Hello` message to the server. The server chooses a ciphersuite and responds with two protocol messages, `A` and `B`, to establish a session key for this ciphersuite. The client completes the handshake by sending a `Finished` message to confirm knowledge of the session key. At the end of the handshake, both the client and server can be sure that they have the same key and that they agree on the ciphersuite. Now suppose we wish to support a new ciphersuite, such that the client receives a different pair of messages, `C` and `D`, between `Hello` and `Finished`. To reuse our well-tested code for processing `Hello` and `Finished`, it is tempting to extend the client state machine to receive either `A` or `C`, followed by either `B` or `D`. This naive composition implements both ciphersuites, but it also enables unintended sequences, such as `Hello; A; D; Finished`. In TLS, clients and servers authenticate the full message sequence at the end of the protocol (in the `Finished` messages) and, since no honest server would send `D` after `A`, allowing extra sequences at the client may seem harmless.

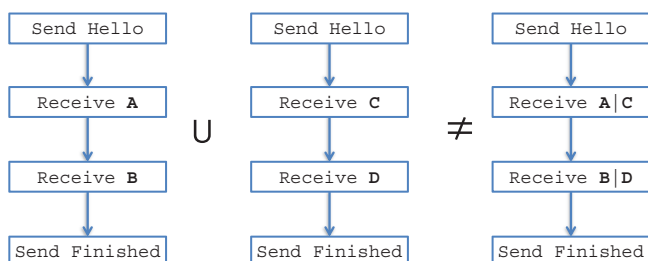
However, a client that accepts this message sequence is actually running an unknown handshake protocol, with *a priori* no security guarantees. In our example, the code for processing `D` expects to run after `C` has been received. If `C` contains the server's signature, then accepting `D` without `C` may allow a crucial authentication step to be bypassed. Furthermore, the code for processing `D` may accidentally use memory that should have been initialized while processing `C`. Such memory safety bugs can lead to dangerous attacks such as HeartBleed,^a which exposed the server's internal state and private keys to remote attackers.

1.2. Testing for state machine flaws

In Section 2, we describe a methodology for systematically

^a <https://heartbleed.com>.

Figure 2. Unsafe composition of two protocol state machines.



testing whether a TLS client or server correctly implements the protocol state machine. We find that many popular TLS implementations exhibit composition flaws like those described above, and consequently accept unexpected message sequences. While some flaws are benign, others lead to critical vulnerabilities that a network attacker can exploit to bypass TLS security. In Section 3, we show how a network attacker can impersonate a TLS server to a buggy client, either by simply skipping messages (SKIP) or by factoring the server's export-grade RSA key (FREAK). These attacks were responsibly disclosed and have led to security updates in major web browsers, servers, and TLS libraries.

1.3. Formally verifying state machine code

We have seen that the security of a TLS implementation depends crucially on its correct implementation of the protocol state machine. Testing can help find some bugs, but once these have been fixed, how can we be sure that the code does not have other hidden flaws? We advocate the use of formal verification to prove the absence of any state machine flaws. In Section 4, we present a new state machine implementation for OpenSSL that supports all commonly enabled ciphersuites, versions, and extensions. Using Frama-C,¹¹ we verify that our code conforms with a logical specification of the TLS protocol state machine.

1.4. Online materials

We refer to <https://mitls.org/pages/attacks/SMACK> for additional details, including security testing tools, a summary of vulnerability disclosures and security updates, our state machine code for OpenSSL, and related verification work on TLS.

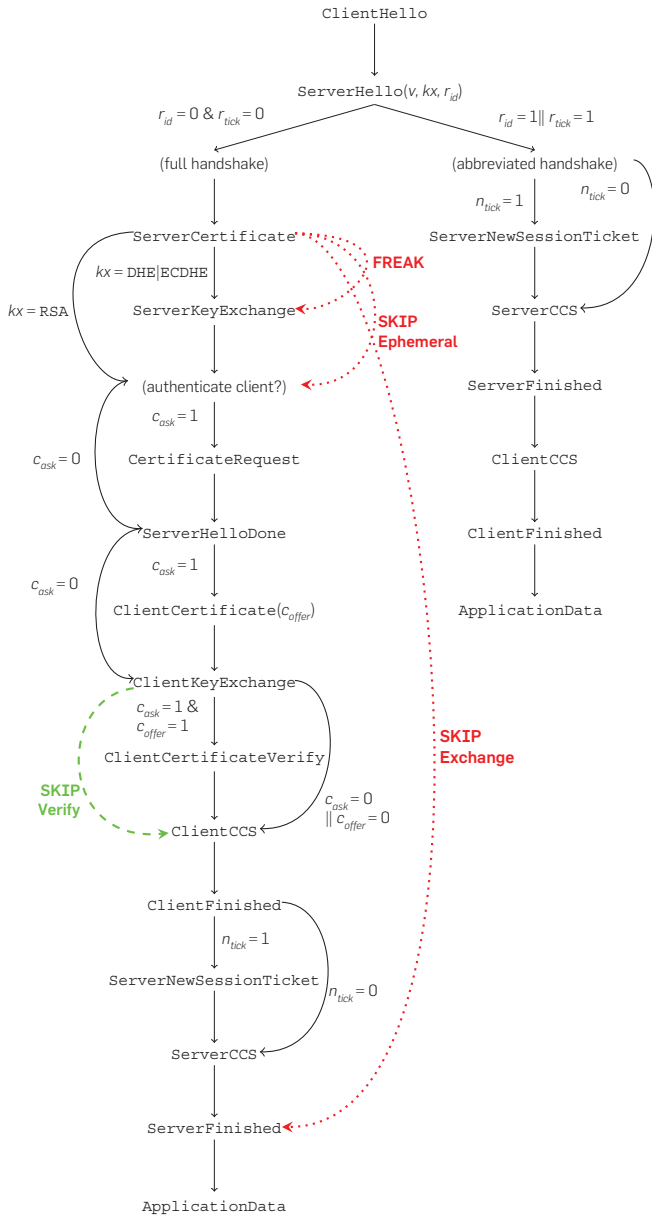
2. TESTING THE TLS STATE MACHINE

The TLS standard¹⁴ does not define a state machine. Instead, it specifies a collection of message sequences, one for each handshake protocol mode. Other specifications add new ciphersuites, authentication methods, or protocol extensions; they typically define their own message sequences, reusing the message formats and mechanisms of TLS, and it is left to the implementation to design a state machine that can account for all these sequences.

2.1. A TLS state machine

We propose a reference state machine for TLS by adopting and extending the one used in the MITLS verified implementation,⁶ based on a careful reading of the standard. Figure 3 depicts a simplified version of this state machine, which can be read from the viewpoint of the client or the server. Each state refers to the last message sent or received; messages prefixed by `Client` are sent by the client; those prefixed by `Server` are sent by the server. Transitions, shown as black arrows, indicate the order in which these messages are expected. When two transitions are possible, each is labeled by the condition under which it is allowed. (Dotted arrows are flawed transitions; they will be explained in Section 3.) The state machine depicted here covers the common usages of TLS on the web, a small but important subset of the full protocol. The figure only shows message sequences; it does not detail message contents, local states, or cryptographic

Figure 3. State machine for commonly used TLS configurations. Paths in the graph represent valid message sequences. Each node indicates the last message sent or received. Black arrows indicate the order in which these messages are expected; labels specify conditions under which the transition is allowed. Dashed arrows on the left show example incorrect transitions found in mainstream TLS servers; dotted arrows on the right show incorrect transitions found in TLS clients. The executed message sequence depends on the negotiated protocol version $v \in \{\text{TLSv1.0}, \text{TLSv1.1}, \text{TLSv1.2}\}$; key exchange $kx \in \{\text{RSA}, \text{DHE}, \text{ECDHE}\}$, and optional features such as fast session resumption (r_{id}, r_{tick}), client authentication (c_{ask}, c_{offer}), and session tickets (n_{tick}).



computations.

Each TLS connection begins with either a full handshake or an abbreviated handshake (also called a resumption). Full handshakes consist of four message flights: the client first sends a `ClientHello`; the server responds with a series of messages from `ServerHello` to `ServerHelloDone`; the

client then sends messages ending with `ClientFinished`; and the server completes the handshake by sending messages ending with `ServerFinished`. The `ServerHello` includes negotiated protocol parameters, such as the version (v) and key exchange method (kx), that determine the rest of the handshake. The `Finished` messages include transcripts of all prior handshake messages, authenticated using the new keys established by the handshake. Before sending them, the client and the server send a change cipher spec (CCS) message to signal the use of the new keys. Once the handshake is complete, the client and the server may exchange streams of `ApplicationData` messages.

The server is identified by a certificate sent in the `ServerCertificate` message. In ephemeral Diffie-Hellman handshakes, the server proves knowledge of the certificate's private key by signing the `ServerKeyExchange` that carries its Diffie-Hellman public key. In the RSA handshake, instead, it uses the private key to decrypt the `ClientKeyExchange` message. In both key exchanges, upon receiving a `CertificateRequest` from the server, the client may optionally send a `ClientCertificate` and use its private key to sign the message transcript so far in the `ClientCertificateVerify`. The state variables c_{ask} and c_{offer} track whether client authentication was requested and accepted.

Abbreviated handshakes rely on shared secrets established in a previous full handshake. The server may store secrets either in a server-side cache or in a client-side session ticket. The server sends a `ServerHello` indicating which kind of resumption will be used, and then goes straight to `ServerCCS` and `ServerFinished`. The client immediately completes the handshake by sending its `ClientCCS` and `ClientFinished` messages.

To what extent does our reference state machine correspond to the one implemented in any given TLS library? For mTLS, we have a type-based proof that its code conforms to this state machine. Next, we investigate mainstream implementations, such as OpenSSL, by systematically testing for deviations from our state machine.

2.2. Generating deviant traces

We first describe the message sequences we use to search for state machine bugs. Let σ be a sequence of messages, m a message, and $\sigma; m$ their concatenation. We write $\sigma \leq \tau$ when σ is a prefix of τ . We write $m \sim m'$ when m and m' have the same message type, but different parameters; for instance when both are `ServerHello` messages selecting different ciphersuites. We also lift \sim from messages to traces. Let V be the set of prefixes of valid traces allowed by the state machine outlined in Figure 3. A *deviant trace* is a minimal invalid trace: $\sigma; m$ is deviant when $\sigma \in V$ but $\sigma; m \notin V$.

Deviant traces are useful for systematically detecting bugs, because a compliant implementation should accept σ and then reject m . If it accepts m , it has a bug. This does not necessarily mean that it has an exploitable security vulnerability: an exploit may involve several carefully crafted messages after the deviant trace. Hence, once we identify an implementation accepting a deviant trace, we look into its source code to learn more about the

cause of the state machine bug.

The set of deviant traces is large (and even infinite unless we bound the number of renegotiations allowed), so we automatically generate a representative, finite subset using three heuristic rules:

Skip If $\sigma; m; n \in V$ and $\delta = \sigma; n \notin V$, test δ . Thus, for every prefix of a valid sequence, we skip a message if it is mandatory. For example, `ClientHello; ServerHello(kx=DHE); ServerKeyExchange` is a trace that skips the `Certificate` message. (Pragmatically, we also skip several messages within flights, but not their last messages, as otherwise the peer is deadlocked.)

Hop Let $\sigma; m \in V$ and $\sigma'; n \in V$. If $\sigma \sim \sigma'$, $m \neq n$, and $\delta = \sigma; n \notin V$, test δ . Thus, if two valid traces have the same prefix, up to their parameters, and they differ on their next messages, we create a deviant trace from the context of the first trace and the next message of the second trace. For example, `ClientHello; ServerHello(kx=RSA); Certificate; ServerKeyExchange` is a trace that sends an unexpected `ServerKeyExchange` by hopping from RSA to Diffie–Hellmann key exchanges.

Repeat If $\sigma; m; \sigma' \in V$ and $\delta = \sigma; m; \sigma'; m \notin V$, test δ . Thus, we resend any message that appears in a valid trace at any subsequent invalid position. For example, `ClientHello; ServerHello; ...; ServerHello-Done; ClientHello` is a trace where the `ClientHello` message is repeated in the middle of a handshake, making it invalid.

An advantage of generating deviant traces from these rules is that, when a trace is accepted by an implementation, it is relatively simple to track the corresponding state machine bug by manual code review. We also experimented with randomly generated deviant traces, but their manual interpretation was more time-consuming and hence less effective.

2.3. Running deviant traces with FlexTLS

As can be expected, generating arbitrary sequences of well-formed messages is hard. By design, each message in a protocol depends on previously exchanged values, and must pass many basic checks before being accepted by the state machine—after all, TLS implementations are meant to comply with the protocol. At the very least, we need to provide reasonable defaults for any missing values, for instance when keys are needed to format a message and yet the peer's input to the key derivation is not available yet.

To this end, we develop FLEXTLS, a tool for scripting and prototyping plausible TLS message sequences. To send and receive messages, FLEXTLS relies on MITLS. Using this robust, verified TLS library helped us to significantly reduce false positives due, for instance, to malformed messages or incorrect cryptographic processing.

FLEXTLS promotes a succinct and purely functional

state-passing style, where each line of code typically corresponds to a message being sent or received. Sending messages out-of-order is as simple as reordering lines in the script. FLEXTLS handles most of the complexity internally, filling in reasonable defaults for any missing values. For example, if the script sends a `Finished` message immediately after a `ServerHello` message, bypassing the full handshake, FLEXTLS would still derive default well-formed connection keys based on empty key exchange values (see Ref.³ for more detailed examples of FLEXTLS scripts).

For each deviant trace, we generate a FLEXTLS client or server script that tests its peer by executing the message sequence, which ends by sending a deviant message. According to the standard, the peer should then send an alert (usually `unexpected_message`) and close the connection. If a non-alert message is received, or the peer does not respond, we assume it wrongly accepted the message, and we flag the trace for further investigation. Not all the TLS implementations we tested support all the scenarios and ciphersuites considered in our traces, and some had unusual error behavior, so we instrumented our scripts to automatically classify peer behavior as correct, unsupported, or wrong. For flagged traces, we manually reviewed the code of the TLS peer, and wrote more detailed FLEXTLS scripts by hand to expose and exploit the state machine flaw.

3. IMPLEMENTATION FLAWS AND ATTACKS

Using FLEXTLS, we tested several mainstream open-source TLS clients and servers for state machine flaws. To ensure maximal support across implementations, we restricted our tests to use TLS 1.0 with RSA and DHE ciphersuites. Table 1 summarizes our experimental results for OpenSSL, GnuTLS, NSS, SecureTransport, Java, Mono, and CyaSSL. Of these, OpenSSL is widely used on servers and on Android phones; NSS is used in many web browsers including Firefox and some versions of Chrome and Opera; SecureTransport is used on Apple devices. Mono and CyaSSL do not support DHE key exchanges, so they are tested on a smaller set of deviant traces. CyaSSL and SecureTransport sometimes tear down the TCP connection when they reject a message, instead of sending a fatal alert as prescribed in the standard, so we filtered out

Table 1. Running deviant traces against mainstream TLS implementations

Library	Key exchange	Traces	Bugs
OpenSSL 1.0.1j	Client RSA, DHE	83	3
	Server RSA, DHE	94	6
GnuTLS 3.3.9	Client RSA, DHE	83	0
	Server RSA, DHE	94	2
SecureTransport 55471.14	Client RSA, DHE	83	3
NSS 3.17	Client RSA, DHE	83	9
Java 1.8.0_25	Client RSA, DHE	71	6
	Server RSA, DHE	94	46
Mono 3.10.0	Client RSA	35	32
	Server RSA	38	34
CyaSSL 3.2.0	Client RSA	41	19
	Server RSA	47	20

such results, and only counted the traces that expose real state machine bugs.

Each bug found by our method corresponds to an unexpected transition in the state machine. For example, Figure 3 shows four bugs we found in various libraries. Extra transitions allowed by clients are depicted as dotted arrows on the right, and those allowed by servers as dotted arrows on the left. Not all such transitions lead to attacks, but in the rest of this section we show how these four transitions can be exploited by an attacker to break the core security guarantees of TLS.

3.1. SKIP exchange (server impersonation)

Our first vulnerability enabled a network attacker to attack TLS clients that used the Java, CyaSSL, or Mono libraries. Our tests found that these client libraries were willing to accept handshakes where the server skips the `ServerCCS` message, thereby disabling encryption for incoming application data. While this is clearly an implementation flaw, it cannot be exploited in isolation; it only becomes an attack when it is combined with a second bug. We also found that Java and CyaSSL clients allowed the server to skip the `ServerKeyExchange` message in Diffie–Hellman exchanges. Since this message normally contains a signature for server authentication, by skipping it, a network attacker can impersonate any server.

Suppose a Java client C wants to connect to some trusted server S (e.g., PayPal). A network attacker M can hijack the TCP connection and impersonate S , without any actual interaction with S , by sending S 's certificate, skipping all messages, notably `ServerKeyExchange` and `ServerCCS`, and directly sending `ServerFinished`. Hence, M bypasses the authenticated key exchange: it can now send unencrypted data to C , and C will interpret it as secure data from S .

Practically exploiting the attack required just a bit more attention to implementation details. The Java and CyaSSL client state machines are so liberal that they allow almost all server messages to be skipped. When they receive the `ServerFinished` message, they authenticate it using an uninitialized master secret (since the key exchange was never performed). The Java client uses an empty master secret, a bytestring of length 0, which M can easily compute. The CyaSSL client compares the received authenticator with an uninitialized block of memory, so M can simply send a bytestring of 12 zeroes, and this will work against any client executed with fresh memory.

In effect, a network attacker can impersonate an arbitrary TLS server S , such as PayPal, to any Java or CyaSSL client. Even if the client carefully inspects the received certificate, it will find it to be perfectly valid for S . Hence, the security guarantees of TLS are completely broken. Furthermore, all the (supposedly confidential and authenticated) traffic between C and M is sent in the clear without any protection.

3.2. SKIP verify (client impersonation)

Our tests showed that OpenSSL, CyaSSL, and Mono allow a malicious client to skip the optional `ClientCertificateVerify` message, even after sending a client certificate to authenticate itself. Since the skipped message normally carries the signature proving ownership of that

certificate, this bug leads to a client impersonation attack, as follows.

Suppose a malicious client M connects to a Mono server S that requires client authentication. M can then impersonate any client C at S by running a regular handshake with S , except that, when asked for a certificate, it provides C 's client certificate instead, and then it skips the `ClientCertificateVerify` message. The server accepts the connection, incorrectly authenticating the client as C , allowing M to read and write sensitive application data belonging to C .

The attack works against Mono as described above, but requires more effort to succeed against other libraries: against OpenSSL, it works only for static Diffie–Hellman certificates, which are rarely used in practice; against CyaSSL, it requires the client to also skip the `ClientCCS` message and then send zeroes in the `ClientFinished` message (like in Section 3.1).

As a result, any attacker can connect to (say) a banking website that uses TLS client certificates to authenticate users. If the website use Mono or CyaSSL, the attacker can login as any user on this website, as long as it knows the user's public certificate. The attack also works if the website uses OpenSSL and allows static Diffie–Hellman certificates.

3.3. SKIP ephemeral (forward secrecy downgrade)

In some settings, a powerful adversary may be able to force a server to reveal its private key (see, e.g., Ref.²⁷) and thus impersonate the server in future connections. Still, we would like to ensure that prior connections to the server (before the private key was revealed) remain secret. This property, commonly called *forward secrecy*, is achieved by the DHE and ECDHE ciphersuites in TLS, whereas RSA, DH, and ECDH ciphersuites do not offer this property.

Forward secrecy is particularly important for web browsers that implement the TLS “False Start” feature.²⁰ These browsers start sending encrypted application data to the server before the handshake is complete. Since the server's chosen ciphersuite (and, in some cases, even the server's identity) has not been authenticated yet, this early application data need the additional protection of forward secrecy.

However, our tests found that NSS and OpenSSL clients allow the server to skip the `ServerKeyExchange` message even in DHE and ECDHE handshakes, which require this message. In such cases, these clients try to use the static key provided in the server certificate as key exchange value, thereby falling back to the corresponding DH and ECDH ciphersuites, without forward secrecy.

Suppose a client based on NSS C (such as Firefox) connects to a website S authenticated by an ECDSA certificate (such as Google) using an ECDHE ciphersuite. A network attacker M can suppress the `ServerKeyExchange` message from S to C . The client then computes the session secrets using the static elliptic curve key of the server certificate, but still believes it is running ECDHE with forward secrecy, and immediately start sending sensitive application data (such as cookies or passwords) because of False Start. Although the connection never completes (as the client and server detect the message suppression at the end of the handshake), the attacker can capture this False Start encrypted data. As a result, assuming

it eventually obtains the server's private key, the attacker will be able to decrypt this data, thereby breaking forward secrecy.

3.4. HOP to RSA_EXPORT (server impersonation)

In compliance with US export regulations before 2000, SSL and TLS 1.0 include several ciphersuites that deliberately use weak keys and are marked as eligible for export. For example, several RSA_EXPORT ciphersuites require that servers send a ServerKeyExchange message with an ephemeral RSA public key (modulus and exponent) whose modulus does not exceed 512 bits. RSA keys of this size were first factorized in 1999⁹ and with advancements in hardware are now considered broken. In 2000, export regulations were relaxed, and in TLS 1.1 these ciphersuites were explicitly deprecated. Consequently, mainstream web browsers no longer offer or accept export ciphersuites. However, TLS libraries still include legacy code to handle these ciphersuites, and some servers continue to support them. We show that this legacy code causes a downgrade attack from RSA to RSA_EXPORT.

Our tests showed that OpenSSL, SecureTransport, and Mono accepted ServerKeyExchange messages even during regular RSA handshakes, in which such messages should never be sent. Upon receiving this message, the client would fallback to RSA_EXPORT by accepting the (signed) 512-bit RSA key in the message and using it instead of the full-size public key in the server certificate. This flaw leads to a man-in-the-middle attack, called FREAK, depicted in Figure 4.

Suppose a client C wants to connect to a server S using RSA, but the server S still supports some RSA_EXPORT ciphersuites. M intercepts C 's RSA handshake to S and responds to C with S 's certificate. In parallel, M connects to S using RSA_EXPORT and ensures that the client and server nonces on the two connections are the same. Now, M forwards S 's ServerKeyExchange to C and, due to the state machine flaw, C accepts this message and overwrites the server's public key with the weaker 512-bit RSA key in this message. Assuming M can factor this key (to obtain the private exponent), it can compute the connection keys and complete the

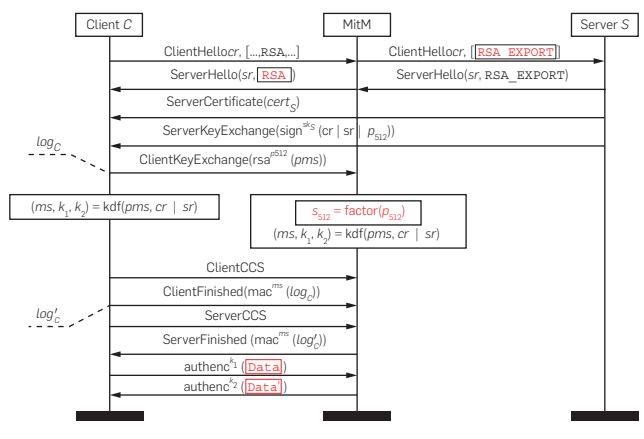
connection, hence impersonating S at C .

FREAK: Factoring 512-bit RSA keys. The main challenge that remains for the attacker is to factor the 512-bit modulus to recover the ephemeral private key during the handshake. First, we observe that 512-bit factorization is now solvable in hours. Second, we note that since computing ephemeral RSA keys on-the-fly can be quite expensive, many implementations of RSA_EXPORT (including OpenSSL) allow servers to precompute, cache, and reuse these public keys for the lifetime of the server (typically measured in days). Hence, the attacker does not need to break the key during the handshake; it can download the key, break it offline, then exploit the attack above for days.

After the disclosure of the vulnerability described above, we collaborated with other researchers to explore its real-world impact. The ZMap team¹⁵ used internet-wide scans to estimate that more than 25% of HTTPS servers still supported RSA_EXPORT, a surprisingly high number. We downloaded the 512-bit ephemeral keys offered by many prominent sites and Nadia Heninger used CADO-NFS^b on Amazon EC2 cloud instances to factor these keys within hours. We then built a proof-of-concept attack demo that showed how a man-in-the-middle could impersonate any vulnerable website to a client that exhibited the RSA_EXPORT downgrade vulnerability. The attack was dubbed FREAK—factoring RSA_EXPORT keys.

We independently tested other TLS implementations for their vulnerability to FREAK. Microsoft SChannel and IBM JSSE also allowed RSA_EXPORT downgrades. Earlier versions of BoringSSL and LibreSSL had inherited the vulnerability from OpenSSL, but they had been recently patched independently of our discovery. In summary, at the time of its disclosure, our server impersonation attack was effective on any client that used OpenSSL, SChannel, SecureTransport, IBM JSSE, or older versions of BoringSSL and LibreSSL. The resulting list of vulnerable clients included most mobile web browsers (Safari, Android Browser, Chrome, BlackBerry, Opera) and a majority of desktop browsers (Chrome, Internet Explorer, Safari, Opera).

Figure 4. FREAK attack: a man-in-the-middle downgrades a connection from RSA to RSA_EXPORT. Then, by factoring the server's 512-bit export-grade RSA key, the attacker can hijack the connection, while the client continues to think it has a secure connection to the server.



3.5. Summary and responsible disclosure

We systematically tested eight TLS libraries including MITLS, found serious state machine flaws in six of them, and were able to mount ten practical attacks, including eight impersonation attacks that break the core security guarantees of TLS.

Almost all implementations allowed some handshake messages to be skipped even if they were required for the current key exchange. We believe that this misbehavior results from a naive composition of handshake state machines. Notably, several implementations allowed CCS messages to be skipped. Considering our attacks as well as the recent Early CCS attack on OpenSSL,^c we note that the handling of CCS messages in TLS state machines is particularly error-prone and deserves close attention. Many implementations (OpenSSL, Java, Mono) also allowed messages to be repeated.

^b <http://cado-nfs.gforge.inria.fr/>.

^c <http://ccsinjection.lepidum.co.jp>.

We reported all the bugs presented in this paper to the various TLS libraries. They were acknowledged and several patches were developed in consultation with us. We then reran FLEXTLS to test whether they fixed the state machine bugs. All of the exploitable bugs we found have now been fixed, but other seemingly benign state machine flaws remain unfixed, and deserve closer analysis in future work.

4. A VERIFIED STATE MACHINE FOR OPENSLL

Systematic state-machine testing uncovers dangerous bugs, but does not guarantee that all flaws have been found and eliminated. Instead, it would be valuable to formally prove that a given state machine implementation complies with the TLS standard. Since new ciphersuites and protocol versions are continuously added to TLS implementations, it would be even better if we could set up an automated verification framework that could be maintained and systematically used to prevent regressions.

The MITLS implementation⁶ uses refinement types to verify that its handshake implementation is correct with respect to a logical state machine specification. Furthermore, it establishes a strong security theorem: a TLS connection between a MITLS client and server is a secure channel, unless one of the low-level cryptographic primitives used by the connection is broken. However, it only covers RSA and DHE ciphersuites and only applies to carefully written F# code.

In this section, we investigate whether we could achieve a similar, if less ambitious, verification result for the state machine implemented by the popular OpenSSL TLS library, which is written in C and covers many more protocol versions, extensions, and ciphersuites than MITLS.

4.1. A new state machine for OpenSSL

The client and server state machines in OpenSSL are coded as loops with large *switch* statements, with one case for each message in the protocol. A series of functions implement the individual messages: each *ssl3_send_** function constructs and sends a message; each *ssl3_get_** function receives and processes a message. These functions maintain the current state in a shared *SSL* data structure with about 100 mutable fields.

The state machine code in OpenSSL has evolved over 17 years to incorporate new protocol versions, ciphersuites, and extensions, resulting in surprisingly complex handling of optional messages and subtle dependencies on various state variables. The current structure makes it difficult to verify whether this code conforms to its intended state machine. Indeed, the flaws in Table 1 indicate that it does not.

We propose a new state machine for OpenSSL that makes the allowed message sequences more explicit and easier to verify. In addition to the full *SSL* data structure used by the messaging functions, we maintain a separate *STATE* data structure (see Figure 5) with just the elements that control state transitions: the role (client or server); the protocol version; the key exchange method; the client authentication mode; flags for resumption and renegotiation; the last message received; and the message sequence so far. By default, each element is initially set to a special *UNDEFINED* value.

The core of our state machine is a single function,

Figure 5. A new state machine for OpenSSL: the *STATE* data structure encodes the current state; *ssl3_next_message* encodes allowed transitions.

```
typedef struct state {
  Role role; // r ∈ {Client, Server}
  PV version; // v ∈ {SSLv3, TLSv1.0, TLSv1.1, TLSv1.2}
  KEM kx; // kx ∈ {DH*, ECDH*, RSA*}
  Auth client_auth; // (cask, coffer)
  int resumption; // (rid, rtick)
  int renegotiation; // = 1 if renegotiating
  int ntick; // = 1 if ticket expected

  Msg_type last_message; // previous message type
  unsigned char* log; // handshake messages so far
  unsigned int log_length;
} STATE;

int ssl3_next_message(SSL* ssl, STATE *st,
  unsigned char* msg, int msg_len,
  int direction, unsigned char content_type);
```

ssl3_next_message, which takes as arguments the current *SSL* and *STATE* structures, the next message to send or receive, its direction, and its content type. This function enforces the state machine on all incoming and outgoing messages. For incoming messages, it checks that the transition is enabled, and then calls the corresponding message handler in legacy code; that code may in turn send some messages, causing our *ssl3_next_message* function to be called in the outgoing direction. For outgoing messages, it similarly checks that the transition is enabled and then calls the usual OpenSSL *ssl_send_** functions.

Our state machine is coded in about 500 lines, supplemented by about 250 lines of simple message parsing functions that can extract message types, protocol versions, and key exchange methods, from various handshake messages.

4.2. Experimental evaluation

To test our new state machine, we deployed it as an inline reference monitor alongside the legacy OpenSSL state machine. Our function *ssl3_next_message* is called before sending or receiving any message, but it does not itself call any message handlers. Instead, it maintains the *STATE* data structure and logs whether the next message violates the state machine. We use this variant of OpenSSL in two ways. First, by running standard interoperability tests for against peers running OpenSSL and other TLS implementations, we check that our new code does not reject valid message sequences. Using this method, we found and fixed some early bugs in our state machine. Second, by running it against deviant FLEXTLS peers, we check that our code logs an error for all the deviant traces presented in Section 2.

4.3. Formal verification

To gain further confidence in our state machine, we formalize our reference TLS state machine as an inductive predicate *isValidState* over the current *STATE* structure. The predicate holds if and only if the message sequence seen so

far is allowed by the state machine. We then specify that this predicate must be maintained as an invariant by our `ssl3_next_message` function.

To mechanically verify that our state machine implementation complies with its `isValidState` specification, we use the C verification tool Frama-C.¹¹ We annotate our code with logical assertions and requirements in Frama-C's specification language, called ACSL, including 460 lines of first-order logic to define `isValidState`. To verify our state machine code, we ran Frama-C to generate proof obligations for multiple SMT solvers. We used Alt-Ergo to discharge some obligations and Z3 for others, for a total verification time of 30 min. Technically, verification also involves memory invariants, to ensure that our code maintains separation between its private state and the rest of OpenSSL, and 900 lines of lemmas to facilitate the proof. (We formally assume that the rest of OpenSSL does not interfere with our code; verifying their full codebase is well beyond the scope of this work.)

4.4. Discussion

Predicates such as `isValidState` are logical encodings of our state machines. They are inspired by the simpler log predicates used in the cryptographic verification of MITLS.⁶ The properties they capture depend only on the TLS specification; they omit any implementation details, and are even independent of their programming languages.

Although our logical specification is almost as long as the code we verified, we found verification useful in several ways. First, in addition to our state invariant, we prove memory safety for our code, a mundane but important goal for C programs. Second, our predicates provide an independent specification of the state machine, and verifying that they agree with the code helped us find bugs, especially regressions due to the addition of new features to the machine. Third, our logical formulation of the state machine allows us to prove theorems about its precision. For example, we used the Coq proof assistant to formally establish that the message sequence stored in `STATE` is unambiguous, that is, if the sequences in two valid states are the same, then the rest of the states must be the same as well. This property is a key lemma for proving the security of TLS, inasmuch as the message transcripts (not the states they encode) are authenticated at the end of the handshake.

Still, our verification result is far from a MITLS-style security theorem for OpenSSL. We proved that our state machine for OpenSSL is functionally correct, but we did not, for example, verify the cryptographic constructions or the full message processing code. We could attempt to extend our results to a larger fragment of OpenSSL that implements all important protocol features; verifying all this code may be feasible but remains a daunting task.

An intermediate goal may be to verify the code in OpenSSL for a single strong ciphersuite, such as `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`. We would then need to prove that, no matter which other ciphersuites are supported, if the client and server choose this ciphersuite, then the resulting connection is secure. To achieve even this limited security theorem, we must overcome several challenges. The first step, which we have already accomplished, is to prove that the state machine correctly implements the message sequence for this

ciphersuite, and that this sequence cannot be confused with that of another ciphersuite. The second step is to prove that it is safe to share the long-term signing keys used in our ciphersuite with other, unverified ciphersuites. This property is problematic for current versions of TLS, but is expected to hold for TLS 1.3.¹⁴ The third step is to show that the session secrets of our verified ciphersuite are cryptographically independent from any other ciphersuite. This property should hold for connections that use TLS 1.3, and also for those that use the TLS extended master secret extension.⁴

In summary, by verifying its state machine, we have taken a first step toward an OpenSSL security theorem, but many problems remain before we can verify mainstream libraries that include legacy code, insecure ciphersuites, and obsolete protocol versions. Partly as a result of our work, the state machine in next major version OpenSSL 1.1.0 was rewritten from scratch, with the goal of making it simpler, stricter, and easier to validate. We hope that with similar efforts in the rest of the codebase, all of OpenSSL will one day become amenable to formal verification.

5. RELATED WORK

5.1. TLS attacks

The reader is advised to refer to Soghoian and Stamm²⁴ for a broad survey of previous attacks on TLS and its implementations; here, we discuss here only closely related work.

Wagner and Schneier²⁸ describe various attacks against SSL 3.0, and their analysis has proved prescient for many attacks on TLS, including the state machine flaws discussed in this paper. For instance, they present an early cross-ciphersuite attack (predating²³) that rely on confusing ephemeral RSA handshakes with ephemeral Diffie–Hellman. They also anticipate some of our message skipping attacks by pointing out that, in MAC-only ciphersuites, the attacker can bypass authentication by skipping CCS messages.

In parallel with our work, de Ruiter and Poll¹² apply machine learning techniques to reverse engineer the state machines of several TLS libraries and discover flaws like the ones described in this paper. Their technique is able to reconstruct abstract state machines even for closed-source libraries, whereas our method focuses on testing conformance to the standard and uncovering concrete exploits.

Jager et al.¹⁷ identify a class of backwards compatibility attacks on protocol implementations that support both strong and weak algorithms, showing for instance how a side-channel attack on RSA decryption in TLS servers can be exploited to mount a cross-protocol attack on server signatures.¹⁸ FREAK, our downgrade attack on export RSA ciphersuites, can also be seen as a backwards compatibility attack. Inspired by FREAK, Logjam¹ is a downgrade attack that exploits a protocol-level ambiguity between the DHE and export DHE ciphersuites. Whereas FREAK relied on a state machine flaw, Logjam relies on the widespread acceptance of weak Diffie–Hellman groups in TLS clients.

Another class of TLS vulnerabilities stems from the incorrect composition of TLS sub-protocols for renegotiation,²⁶ alerts,⁶ and resumption.⁸ These flaws may be partly blamed on the state machine being underspecified in the standard—the last two were discovered while designing and verifying the state machine of MITLS.

5.2. TLS verification

Cryptographers have developed proofs for DHE,¹⁶ RSA,¹⁹ and PSK²² key exchanges run in isolation; they apply to the TLS design, but not its implementations.

Bhargavan et al.^{6, 7} proved that composite RSA and DHE are jointly secure in the mTLS implementation, programmed in F# and verified using refinement types.

Several works extract formal models from TLS implementations and analyze them with automated protocol verification tools. Bhargavan et al.⁵ extract and verify ProVerif and CryptoVerif models from an F# implementation of TLS. Chaki and Datta¹⁰ verify the SSL 2.0/3.0 handshake of OpenSSL using model checking and find several known rollback attacks. Avalle et al.² verify Java implementations of the TLS handshake protocol using ProVerif.

Others analyze TLS libraries for programming bugs. Lawall et al.²¹ use the Coccinelle framework to detect incorrect checks on values returned by the OpenSSL API, and Frama-C has been used to verify parts of PolarSSL.


6. CONCLUSION

While security analyses of TLS primarily focused on flaws in fixed cryptographic constructions, the state machines that control the flow of protocol messages in their implementations have escaped scrutiny. Using a combination of automated testing and manual source code inspection, we discovered serious flaws in several TLS implementations. These flaws predominantly arise from the incorrect composition of the multiple ciphersuites and authentication modes supported by TLS.

Considering the impact and prevalence of these flaws, we advocate a principled programming approach for protocol implementations that includes systematic testing against unexpected message sequences (a form of directed fuzzing) as well as formal proofs of correctness for critical components.

Although current TLS implementations are far from perfect, upcoming improvements in the protocol and progress in verification tools let us hope that the security verification of mainstream TLS libraries will soon be within reach.

Acknowledgments

The authors would like to thank Matthew Green, Nadia Heninger, Santiago Zanella-Béguelin, the ZMap team, and the CADO-NFS team for their help with evaluating and exploiting FREAK. We thank the developers of OpenSSL, SChannel, SecureTransport, NSS, BoringSSL, Oracle JSSE, CyaSSL, and Mono for their rapid response to our disclosures. Bhargavan, Beurdouche, and Delignat-Lavaud were supported by the ERC Starting Independent Researcher Grant no. 259639 (CRYSP). 

References

1. Adrian, D., Bhargavan, K., Durumeric, Z., Gaudry, P., Green, M., Halderman, J.A., Heninger, N., Springall, D., Thomé, E., Valenta, L., VanderSloot, B., Wustrow, E., Zanella-Béguelin, S., Zimmermann, P. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *ACM CCS* (2015), 5–17.
2. Avalle, M., Pironti, A., Pozza, D., Sisto, R. JavaSPI: A framework for security protocol implementation. *Int. J. Sec. Softw. Eng.* 2 (2011), 34–48.
3. Beurdouche, B., Delignat-Lavaud, A., Kobeissi, N., Pironti, A., Bhargavan, K. FlexTLS: A tool for testing TLS implementations. In *USENIX Workshop on Offensive Technologies*

- (WOOT) (2015).
4. Bhargavan, K., Delignat-Lavaud, A., Pironti, A., Langley, A., Ray, M. Transport Layer Security (TLS) session hash and extended master secret extension. IETF RFC 7627, 2014.
5. Bhargavan, K., Fournet, C., Corin, R., Zălinescu, E. Verified cryptographic implementations for TLS. *ACM TISSEC* 15, 1 (2012), 1–32.
6. Bhargavan, K., Fournet, C., Kohlweiss, M., Pironti, A., Strub, P. Implementing TLS with verified cryptographic security. In *IEEE S&P (Oakland)* (2013), 445–459.
7. Bhargavan, K., Fournet, C., Kohlweiss, M., Pironti, A., Strub, P.-Y., Zanella-Béguelin, S. Proving the TLS handshake secure (as it is). In *CRYPTO* (2014), 235–255.
8. Bhargavan, K., Lavaud, A.D., Fournet, C., Pironti, A., Strub, P.-Y. Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS. In *IEEE S&P (Oakland)* (2014), 98–113.
9. Cavallar, S., Dodson, B., Lenstra, A., Lioen, W., Montgomery, P., Murphy, B., te Riele, H., Aardal, K., Gilchrist, J., Guilleme, G., Leyland, P., Marchand, J., Morain, F., Muffett, A., Putnam, C., Zimmermann, P. Factorization of a 512-bit RSA modulus. In *EUROCRYPT* (2000), 1–18.
10. Chaki, S., Datta, A. ASPIER: An automated framework for verifying security protocol implementations. In *IEEE CSF* (2009), 172–185.
11. Cuoq, P., Kirchner, F., Kosmatov, N., Prevosto, V., Signoles, J., Yakobowski, B. Frama-C. In *Software Engineering and Formal Methods* (2012), 233–247.
12. de Ruitter, J., Poll, E. Protocol state fuzzing of TLS implementations. In *USENIX Security* (2015), 193–206.
13. Dierks, T., Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.2. IETF RFC 5246, 2008.
14. Dierks, T., Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.3. Internet Draft, 2014.
15. Durumeric, Z., Wustrow, E., Halderman, J.A. ZMap: Fast Internet-wide scanning and its security applications. In *USENIX Security* (2013), 605–620.
16. Jager, T., Kohlar, F., Schäge, S., Schwenk, J. On the security of TLS-DHE in the standard model. In *CRYPTO* (2012), 273–293.
17. Jager, T., Paterson, K.G., Somorovsky, J. One bad apple: Backwards compatibility attacks on state-of-the-art cryptography. In *NDSS* (2013).
18. Jager, T., Schwenk, J., Somorovsky, J. On the Security of TLS 1.3 and QUIC Against Weaknesses in PKCS#1 v1.5 Encryption. In *ACM CCS* (2015), 1185–1196.
19. Krawczyk, H., Paterson, K.G., Wee, H. On the security of the TLS protocol: A systematic analysis. In *CRYPTO* (2013), 429–448.
20. Langley, A., Modadugu, N., Moeller, B. Transport Layer Security (TLS) False Start. IETF RFC 7918, 2010.
21. Lawall, J., Laurie, B., Hansen, R.R., Palix, N., Muller, G. Finding error handling bugs in OpenSSL using Coccinelle. In *European Dependable Computing Conference* (2010), 191–196.
22. Li, Y., Schäge, S., Yang, Z., Kohlar, F., Schwenk, J. On the security of the pre-shared key ciphersuites of TLS. In *Public-Key Cryptography* (2014), 669–684.
23. Mavrogianopoulos, N., Vercauteren, F., Velichkov, V., Preneel, B. A cross-protocol attack on the TLS protocol. In *ACM CCS* (2012), 62–72.
24. Meyer, C., Schwenk, J. Lessons learned from previous SSL/TLS attacks – A brief chronology of attacks and weaknesses. IACR Cryptology ePrint Archive, Report 2013/049, 2013.
25. Paterson, K.G., Ristenpart, T., Shrimpton, T. Tag size does matter: Attacks and proofs for the TLS record protocol. In *ASIACRYPT* (2011), 372–389.
26. Ray, M., Dispensa, S. Renegotiating TLS, 2009.
27. Soghoian, C., Stamm, S. Certified lies: Detecting and defeating government interception attacks against SSL. In *Financial Cryptography* (2012), 250–259.
28. Wagner, D., Schneier, B. Analysis of the SSL 3.0 protocol. In *USENIX Workshop on Electronic Commerce* (1996), 29–40.

Benjamin Beurdouche and Karthikeyan Bhargavan ({benjamin.beurdouche, karthikeyan.bhargavan, antoine.delignat-lavaud, alfredo.pironti}@inria.fr), INRIA.

Alfredi Pironti (alfredo@pironti.eu), IOActive.

Antoine Delignat-Lavaud, Cédric Fournet, and Markulf Kohlweiss ({antdl, fournet, markulf}@microsoft.com), Microsoft Research.

Pierre-Yves Strub (pierre Yves.strub@imdea.org), IMDEA Software Institute.

Jean-Karim Zinzindohoué (jean-karim.zinzindohoue@inria.fr), INRIA & Ecole des Ponts, ParisTech.

Authentication Using Pulse-Response Biometrics

By Ivan Martinovic, Kasper Rasmussen, Marc Roeschlin, and Gene Tsudik

Abstract

We propose a new biometric based on the human body's response to an electric square pulse signal, called *pulse-response*. We explore how this biometric can be used to enhance security in the context of two example applications: (1) an additional authentication mechanism in PIN entry systems, and (2) a means of continuous authentication on a secure terminal. The pulse-response biometric is effective because each human body exhibits a unique response to a signal pulse applied at the palm of one hand, and measured at the palm of the other. Using a prototype setup, we show that users can be correctly identified, with high probability, in a matter of seconds. This identification mechanism integrates well with other established methods and offers a reliable additional layer of security, either on a continuous basis or at login time. We build a proof-of-concept prototype and perform experiments to assess the feasibility of pulse-response as a practical biometric. The results are very encouraging, achieving accuracies of 100% over a static data set, and 88% over a data set with samples taken over several weeks.

1. INTRODUCTION

Many modern access control systems augment the traditional two-factor authentication procedure (something you know and something you have) with a third factor: “something you are,” that is, some form of biometric authentication. This additional layer of security comes in many flavors: from fingerprint readers on laptops used to facilitate easy login with a single finger swipe, to iris scanners used as auxiliary authentication for accessing secure facilities. In the latter case, the authorized user typically presents a smart card, then types in a PIN, and finally performs an iris (or fingerprint) scan.

In this paper, we propose a new biometric based on the human body's response to a square pulse signal. We consider two motivating scenarios:

The first is the traditional access control setting described above where the biometric is used as an additional layer of security when a user enters a PIN, for example, into a bank ATM. The pulse-response biometric facilitates unification of PIN entry and biometric capture. We use PIN entry as a running example for this scenario throughout the paper. This is because PIN pads are often made of metal, which makes capturing pulse-response biometric straightforward: a user would place one hand on a metal pad adjacent to the key-pad, while using the other hand to enter a PIN. This conductive pad would transmit the pulse and a sensor in the PIN pad would capture the measurement.

The second scenario corresponds to *continuous authentication*, for example, verifying that the user, who securely logged in earlier, is the same person currently present at the keyboard. For this scenario, we need a mechanism that periodically samples one or more biometric. However, for obvious usability reasons, ideally this would be done *unobtrusively*. The pulse-response biometric is particularly well-suited for this setting. Assuming that it can be made from—or coated by—a conductive material, the keyboard would generate the pulse signal and measure response, while the user (remaining oblivious) is typing. The main idea is that the user's pulse-response is captured at login time and the identity of the person currently at the keyboard can be verified transparently, at the desired frequency.

To assess the efficacy and feasibility of the pulse-response biometric, we built a prototype platform that enables gathering pulse-response data. Its main purpose is to assess whether we can identify users from a population of test subjects. The same platform can test the distinguishing ability and stability of this biometric over time. We also explored two systems that apply the pulse-response biometric to the two sample scenarios discussed above: one to unobtrusively capture the biometric as an additional layer of security when entering a PIN, and the other to implement continuous authentication.

2. BACKGROUND

This section provides background on biometrics, summarizes the terminology and introduces our design goals.

2.1. Biometrics

The meaning of *biometric* varies depending on context. Throughout this paper we use it to denote a measurable biological (anatomical and physiological) or behavioral *characteristic* that can be used for automated recognition of individuals.

Usually, biometric measurements are divided into two categories, physiological, and behavioural.³ The former relies on the physiology of a person, such as fingerprints, facial features, or DNA. Behavioral biometrics are based on user behavior, such as keystroke timings, speech patterns, handwriting characteristics, gait, and many others.

Physiological biometrics can help identify an individual among large pool of candidates. However, there are some caveats. In general, physiological biometrics are considered moderately difficult to circumvent. For example, although

A full version of this paper was presented at the *Network and Distributed System Security (NDSS) Symposium 2014*.

hand geometry is very stable over the course of one's adult life, it does not provide enough distinguishing power to be used as the only means for identification.⁷ Also, some facial recognition systems can be fooled by an appropriately sized photo of a legitimate user.

Behavioral biometrics measure user actions over time, that is, for each action, there must be a beginning, an end, and a duration. Consequently, behavioral biometrics indirectly measure characteristics of the human body. Behavioral biometrics are learned and, therefore, can be also re-learned. However, the consensus in the literature seems to be that after reaching a certain age, changes in behavior become more difficult to achieve, even with specific and sustained effort.¹¹ Behavioral biometrics can therefore be regarded as valid means of identification, even though they are neither as unique nor as permanent as their physiological counterparts. In most cases, behavioral biometrics are used to discern a user from a small(er) pool of candidates. One advantage is that they are less invasive and therefore more user-friendly. For example, a system that analyses keystroke timings or speech patterns can usually do so in the background. In contrast, an iris or fingerprint scan requires specific user actions.

2.2. Biometric authentication versus identification

Authentication refers to identify confirmation or verification. When a user claims a certain identity (e.g., by inserting a card into an ATM or entering a user ID into a terminal and then typing in a PIN or a password) authentication entails deciding whether the claim is correct. The goal of the biometric classifier is to compare the current sample to the known template for that user. The classifier returns the likelihood of a match. We refer to this as a 1:1 *comparison*.

Authentication differs from identification, where the current sample comes from an unknown user, and the job of the biometric classifier is to match it to a known sample. We refer to this as a 1:*n comparison*. Identification is further divided into two types: open-set and closed-set. We say that an identification is closed-set, if it is known *a priori* that the user is in the classifier database, that is, the classifier must choose the best match from a pool of candidates. Otherwise, identification is considered open-set.

2.3. Design goals

When designing a new biometric system it is important to take into account lessons learned from past and current systems. Design goals for biometric systems can be found in the literature, for example, Jain et al.⁴ Our goals include, but are not limited to:

Universal. Must be universally applicable, to the extent required by the application. It is important for the biometric to apply to everyone who is intended to use the system.

Unique. Must be unique within the target population. For example, measuring someone's height would not work as an identification mechanism on a large scale. At the same time, (adult) height alone *can* usually identify individual family members.

Permanent. Must remain consistent over the period of use. Very few biometrics will stay constant over a lifetime, for

example, face geometry, voice, gait, and writing. However, as long as the biometric is consistent over the lifetime of the system, these biometrics work well.

Unobtrusive. If the user can be identified passively, without interference, the biometric is much more likely to be accepted.

Difficult to circumvent. Ideally, a user should be unable to change the biometric at all. At a minimum, a user must not be able to modify his biometric to match that of another user.

3. PULSE-RESPONSE BIOMETRIC

The pulse-response biometric works by applying a low voltage pulse signal to the palm of one hand and measuring the body's response in the palm of the other hand. The signal travels up through the user's arm, across the torso, and down the other arm. The biometric is captured by measuring the response in the user's hand. This response is then transformed to the frequency domain via the Fast Fourier Transform (FFT). This transformation yields the individual frequency components (bins) of the response signal, which form raw data that is then fed to a classifier. Working in the frequency domain eliminates any need for aligning the pulses when they are measured.

The main reason for the ability of this biometric to distinguish between users is due to subtle differences in body conductivity, at different frequencies, among different people. When a signal pulse is applied to one palm and measured in the other, the current travels through various types of body tissues—blood vessels, muscle, fat tissue, cartilage, and bones—to reach the other hand. Differences in bone structure, muscle density, fat content, and layout (and size) of blood vessels result in slight differences in the attenuation of the signal at different frequencies. These differences show up as differences in the magnitude of the frequency bins after the FFT. This is what facilitates distinguishing among individuals.

Pulse-response is a physiological biometric since it measures body conductivity—a physiological characteristic distinct from behavioral aspects. However, it has an attractive property normally associated with behavioral biometrics: it can be captured in a completely passive fashion. Although other physiological biometrics also have this feature, for example, face recognition, pulse-response is not easily circumventable. This combination of unobtrusiveness and difficulty to circumvent makes it a very attractive identification mechanism. Essentially, it offers the best properties of both physiological and behavioral biometrics.

4. LIVENESS AND REPLAY

A common problem with many biometric systems is liveness detection, that is, determining whether the biometric sample represents a "live" user or a replay. For example, a fingerprint reader would want to detect whether the purported user's fingerprint was produced by a real finger attached to a human, as opposed to a fingerprint mold made of putty or even a severed finger. Similarly, a face recognition system would need to make sure that it is not being fooled by a user's photo or a 3D replica.

In traditional biometric systems, liveness is usually

addressed via some form of active authentication, for example, a challenge-response mechanism. In a face recognition system a user might be asked to turn his head or look at a particular point during the authentication process. Although this reduces the chance of a photo passing for the real person, the user is forced to take active part in the process, which can be disruptive and annoying if authentication happens on a continuous basis. Also, a good 3D model of a human head can still fool such measures.

Fingerprint scanners often include some protection against replay. This might be accomplished by detecting other characteristics normally associated with a live finger, for example, temperature, or presence of sweat or skin oils. Such counter-measures make it more difficult to use skin-tight gloves or “cold dead fingers” to fool the biometric system. Still, replay remains a major challenge, especially for low-end fingerprint readers.

In the context of the pulse-response biometric, unlike fingerprints or face recognition, it is difficult (yet not impossible) to separate the biometric from the individual to whom it belongs. If the adversary manages to capture a user’s pulse-response on some compromised hardware, replaying it successfully would require specialized hardware that mimics the exact conductivity of the original user. We believe that this is feasible: the adversary can devise a contraption that consists of flat adhesive-covered electrodes attached to each finger-tip (five for each hand going into one terminal) with a single wire connecting the two terminals. The pulse response of the electrode-wire-electrode has to exactly replicate that of the target user. Having attached electrodes to each finger-tip, the adversary can type on the keyboard and the system could thus be effectively fooled. However, the effort required is significantly harder than in cases of facial recognition (where a photo suffices) or fingerprints, which are routinely left—and can be lifted from—numerous innocuous locations.

Finally, the real power of the pulse-response biometric is evident when used for continuous authentication (see Section 6). Here, the person physically uses a secure terminal and constantly touches the keyboard as part of routine work. Authentication happens on a continuous basis and it is not feasible to use the terminal while at the same time providing false input signals to the authentication system. Of course, the adversary could use thick gloves, thereby escaping detection, but the authentication system will see input from the keyboard without the expected pulse-response measurement to accompany it, and will lock the session.

5. COMBINING PIN ENTRY WITH BIOMETRIC CAPTURE

This section describes the envisaged use of pulse-response to unobtrusively enhance the security of PIN entry systems.

5.1. System and adversary models

We use a running example of a metal PIN key-pad with an adjacent metal pad for the user’s other hand. The keypad has the usual digit (0–9) buttons as well as an “enter” button. It also has an embedded sensor that captures the pulse-signal transmitted by the adjacent metal pad. This setup corresponds to a

bank ATM or a similar setting.

The adversary’s goal is to impersonate an authorized user and withdraw cash. We assume that the adversary cannot fool the pulse-response classifier with probability higher than that found in our experiments described later in this paper.

We also assume that the ATM is equipped with a modified authentication module which, besides verifying the PIN, captures the pulse-response biometric and determines the likelihood of the measured response corresponding to the user identified by the inserted ATM card and the just-entered PIN. We assume that the ATM has access to a database of valid users, either locally or over a network. Alternatively, the user’s ATM card can contain data needed to perform pulse-response verification. If stored on the card, this data must be encrypted and authenticated using a key known to the ATM; otherwise, the adversary (who can be assumed to be in possession of the card) could replace it with data matching its own pulse-response.

5.2. PIN entry scheme

The ATM has to determine whether data sampled from the user while entering the PIN is consistent with that stored in the database. This requires a classifier that yields the likelihood of a sample coming from a known distribution. The likelihood is used to determine whether the newly measured samples are close enough to the samples in the database to produce a match. Using our prototype, we can make such decisions with high confidence; see Section 7.4.

Before discussing security of the pulse-response PIN entry system, we check whether it meets the design goals.

Universal. A person using the modified PIN entry system must use both hands, one placed on the metal pad and one to enter the pin. This requires the user to actually have two hands. In contrast, a normal PIN entry system can be operated with one hand. Thus, universality of our system is somewhat lower. This is a limitation of the biometric, although a remedy could be to store a flag on the user’s ATM card indicating that disability, thus exempting this person from the pulse-response check. This would allow our approach to gracefully degrade to a generic PIN entry system.

Unique and Permanent. In Section 7.4, we show that our prototype can determine, with high probability, whether a subject matches a specific pulse-response. Thus, it is extremely unlikely for two people to exhibit exactly the same pulse-response. We also show that an individual’s pulse-response remains fairly consistent over time.

Unobtrusive. The proposed scheme is very unobtrusive, since from the user’s perspective, the only thing that changes from current operation is the added requirement to place the free hand on a metal pad. There can even be two such pads accommodating both left- and right-handed people. Also, the ATM screen could display system usage instructions, even pictorially to accommodate people who cannot read. Similarly, audio instructions could be given for the sake of those who are vision-impaired.

Difficult to circumvent. Given that pulse-response is unique, the only other way to circumvent it is to provide the sensor (built into the PIN pad) with a signal that would correspond

to the legitimate user. Although this is very hard to test precisely, assuming that the adversary is unaware of the target user's pulse-response measurements, the task seems very difficult, if not impossible.

5.3. Security of PIN entry scheme

The additional layer of security provided by the pulse-response biometric is completely independent from security of the PIN entry system alone. Therefore, we model the probability P_{break} that the proposed PIN entry system can be subverted, as:

$$P_{break} = P_{guess} \cdot P_{forge}$$

where P_{guess} is the probability of the adversary correctly guessing the PIN and P_{forge} is the average probability that the adversary can fool the classifier. We model this as the false positive rate divided by the number of users. The false positive rate, that is, when an adversary is incorrectly classified as an authorized user, is the complement of specificity.¹⁰ In Section 7.4, we determine specificity to be 88% and thus $P_{forge} = (1 - 0.88)$ on average.

If a PIN consists of n decimal digits and the adversary has t guesses then $P_{guess} = \frac{t}{10^n}$. Together with P_{forge} this yields the combined probability:

$$P_{break} = \frac{(1 - 0.88)t}{10^n}$$

For example, if the adversary is allowed three guesses with a 4-digit pin, $P_{break} = 3.6 \cdot 10^{-5}$, whereas a 4-digit plain-PIN system has a subversion probability of $3 \cdot 10^{-4}$. Though this improvement might not look very impressive on its own, it is well known that most PIN attacks are performed by "shoulder surfing" and do not involve the adversary guessing the PIN. If we assume that the adversary already knows the PIN, $P_{break} = 12\%$ with our system, as opposed to 100% without it.

6. CONTINUOUS AUTHENTICATION

We now present a continuous authentication scheme. Its goal is to verify that the same user who securely logged into a secure terminal, continues to be physically present at the keyboard. Here, the pulse response biometric is no longer used as an additional layer of security at login time. Rather, the user's pulse-response biometric is captured at login time and subsequent measurements are used to authenticate the user using the initial reference.

6.1. System and adversary models

We continue using the example for continuous authentication introduced in Section 1. It entails a secure terminal where authorized users can login and access sensitive data.

The system consists of a terminal with a special keyboard that sends out pulse signals and captures the pulse-response biometric. This requires the keyboard to be either made from, or coated by, a conductive material. Alternatively, the pulse signal transmitter could be located in a mouse that the user operates with one hand and the keyboard captures the pulse-response. Without loss of generality, we assume the former option.

We assume that the adversary, with or without consent

of the authorized (at login time) user, physically accesses the unattended terminal and attempts to proceed within an already-open session. We assume that the adversary at the keyboard has full access to the active session. The goal of our system is to detect that the original user is no longer present, and that the keyboard is operated by someone else. If a different user is detected, the system consults a policy database and takes appropriate actions, for example, locks the session, logs out the original user, raises alarms, or notifies system administrators.

In addition to the peripherals required to capture the pulse-response signal, the continuous authentication system consists of a software process that manages initial login and frequency of periodic reacquisition of the biometric. This process is also responsible for displaying user warnings and reacting to suspected violations. We refer to it as the *continuous authentication process* (CAP) and assume that neither the legitimate user nor the adversary can disable it.

6.2. Continuous authentication scheme

At login time, CAP measures and records the initial pulse-response biometric of the authorized user. Periodically, for example, every few seconds, CAP reacquires the biometric by sending and receiving a pulse signal through the keyboard. Each newly acquired measurement is checked against the value acquired at login. If the new measurement is sufficiently distinct from that sampled from the original user, CAP consults its policy database and takes appropriate actions, as discussed above. Figure 1 shows a sample CAP decision flowchart.

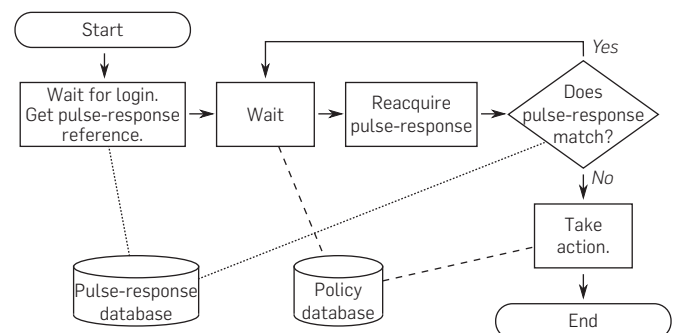
The envisaged continuous authentication system can be useful for training (e.g., corporate) users to adopt security-conscious behavior. For example, users can be motivated to behave securely whenever they leave a secure terminal, for example, by getting a warning every time they forget to log out and/or allow someone else to take over a secure session.

Before considering the security of the continuous authentication system, we look back at the design goals.

Universal. The users of the system must have two hands in order for the pulse-response biometric to be captured. The same arguments, as in the case of PIN entry, apply here.

Unique and Permanent. In Section 7.4, we show that our prototype can match a pulse-response to previous samples

Figure 1. Flowchart of the Continuous Authentication Process decision procedure.



(taken immediately beforehand) with 100% accuracy. The fact that the pulse-response reference is taken at the beginning of the session and is used only during that session, makes it easier to overcome consistency issues that can occur when the reference and test samples are days or months apart.

Unobtrusive. Users do not need to modify their behavior at all when using the continuous authentication system. Thus, user burden is minimal.

Difficult to Circumvent. With a true positive rate of 100% it is unlikely that the adversary can manage to continuously fool the classifier. Even if the adversary happens to have a pulse-response biometric similar to the original user, it must evade the classifier on a continuous basis. We explore this further in the security analysis section below.

6.3. Security

The adversary's goal is to subvert the continuous authentication system by using the secure terminal after the original user has logged in. In the analysis below, we assume that the original user colludes with the adversary. This eliminates any uncertainty that results from the original user "discovering" that the adversary is using its terminal, which is hard to model accurately. This results in a worst-case scenario and the detection probability is a lower bound on security provided by the continuous authentication system.

An important measure of security is the detection time—the number of times biometric acquisition is performed between the adversary's initial appearance and detection. Obviously, longer inter-acquisition intervals imply slower collection of measurements and subsequent detection of adversarial presence.

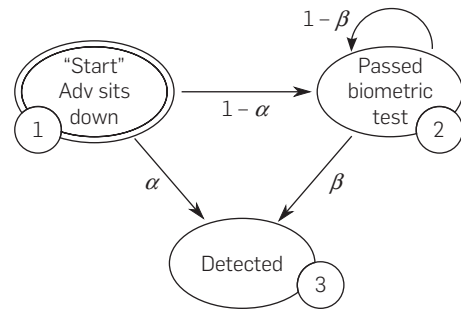
We model the probability of detecting an adversary using two static probabilities derived from our experiments—an initial probability α and a steady state probability β . A more detailed model with several intermediate decreasing probabilities could be constructed but this simple model fits quite well with our experiments.

The probability α is the probability that the adversary is detected immediately, that is, the very first time when his pulse-response is measured. However, if the adversary's biometric is very close to that of the original user, the adversary might not be detected every time biometric capture is performed. This is because the biometric is subject to measurement noise and the measurements from an individual form a distribution around the "fingerprint" of that user. If the adversary manages to fool the classifier once, it must be because its biometric is close to that of the original user. Thus, the adversary's subsequent detection probability must be lower:

$$P[X_i = adv | X_{i-1} = usr] \leq P[X_i = adv]$$

We call this decreased probability β . The probabilities α and β are approximations that model how similar two individuals are, that is, how well their probability distributions overlap in about 100 dimensions. Using α and β we build a Markov model, shown in Figure 2, with three states to calculate the probability that the adversary is detected after i rounds.

Figure 2. Markov model of the continuous authentication detection probability. States are numbered 1–3 for easy reference in text.



When the adversary first accesses the keyboard, it is either detected with probability α or not detected, with probability $1 - \alpha$. In the latter case, its pulse-response biometric must be close to the original user's. Thus, β is used for the subsequent rounds. In each later round, the adversary is either detected with probability β or not detected, with probability $1 - \beta$. To find the combined probability of detection after i rounds, we construct the state transition matrix P of the Markov model, as follows:

$$P = \begin{bmatrix} 0 & 1 - \alpha & \alpha \\ 0 & 1 - \beta & \beta \\ 0 & 0 & 1 \end{bmatrix}$$

Each row and each column in P corresponds to a state. The entry in row q and column r , p_{qr} , is the probability of transitioning from state q to state r . To find the probabilities of each state we start with a row vector ρ that represents the initial probability of being in state 1, 2, and 3. Clearly, $\rho = [1, 0, 0]$, indicating that we always start in state 1. The probability of being in each state after one round (or one transition) can be represented by the inner product ρP . Probabilities for each subsequent round are determined via another multiplication by P . The probabilities of being in each state after i rounds (state transitions), is therefore:

$$[1, 0, 0] \cdot P^i = [0, (1 - \alpha)(1 - \beta)^{i-1}, 1 - (1 - \alpha)(1 - \beta)^{i-1}]$$

As expected, the probability of being in state 1 (the initial state) is 0, since the first state transition forces a transition from the initial state and there is no way back (see Figure 2). The probability of being in state 2, that is, to escape detection for i rounds, is given by the second element of ρ : $(1 - \alpha)(1 - \beta)^{i-1}$. The probability of detection is thus: $1 - (1 - \alpha)(1 - \beta)^{i-1}$.

α roughly corresponds to the *sensitivity* of the classifier, that is, the true positive rate reported in Section 7. We use 99% (rather than the 100% found in our experiments) in order to model the possibility of making a classification mistake at this point. We do not have enough data to say with absolute certainty if this is valid for very large populations, but we continue under the assumption that our data is representative. β is harder to estimate but we set $\beta = 0.3$ based on numbers from our experiments in Section 7.4. Using these values there is a 99.96% chance of detecting the adversary after 10

rounds. This grows to 99.9999997% after 50 rounds. Thus, not surprisingly, acquisition frequency determines the time to detect the adversary.

What the very high 99.999+% detection probability is really saying is that, if you just test enough times, the authentication will eventually fail. It matches very well with our experiments and it is true even for a legitimate user (although much less frequently). For this reason we need a way to handle false negatives.

6.4. Handling false negatives

False negatives refer to incorrect detection of adversarial presence. If the biometric is used as an additional layer of security during the authentication procedure, this can be managed simply by restarting the login procedure, if the first attempt fails. However, in a continuous authentication setting, where a single (and possibly incorrect) detection might cause the system to lock up, false negatives have to be handled more thoughtfully.

One approach is to specify a policy that allows a certain number of detection events every n th round, without taking any action. For example, allowing one event every 100 rounds corresponds to a false negative rate of 1%. Another option is to integrate a less user-friendly (less transparent) biometric to deal with ambiguous detection events. For example, after a few detection events, the user might be asked to confirm his identity by swiping a thumb on a fingerprint scanner.

Yet another alternative is the gradual ramp up of the severity of actions taken by the CAP, for each successive detection event. For the first time, displaying a warning might be the most appropriate action. If detection re-occurs, more and more severe actions can be taken. It is very unlikely, with a reasonably low false negative rate, to have multiple consecutive adversary detection events if the original user is still at the terminal. Although the false positive rates we achieve are quite low, they could certainly be improved with a more advanced biometrics capture system. In conjunction with a sensible policy, our continuous authentication system might be appropriate for any organization with high security requirements.

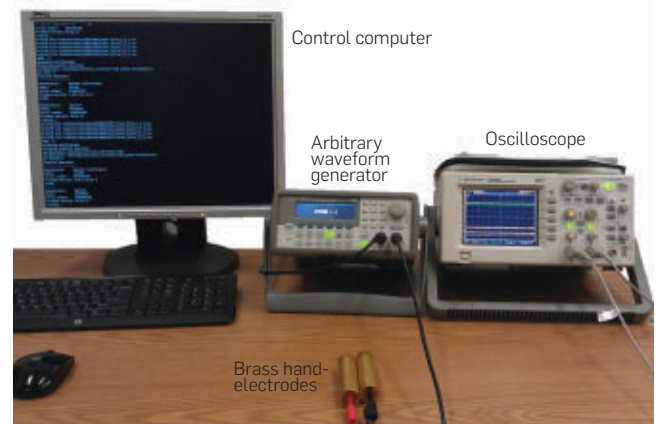
7. EXPERIMENTS

Starting out with the hypothesis that the biometric measurement varies depending on the frequency of the signal transmitted through the human body, we rigorously experimented with various frequencies, voltage levels and waveforms. We also assessed several classification algorithms. Our experiments suggested the choice of 100 ns long square pulses at 1 V as the input signal (see Figure 4) and *Support Vector Machines (SVM)* for classifying samples. Hence, the name *pulse-response* biometric. Complete analysis can be found in the full version of this paper.⁸

7.1. Measurement setup

In order to gather stable and accurate pulse-response measurements we build a data acquisition platform consisting of: (1) an arbitrary waveform generator, (2) an oscilloscope, (3) a pair of brass electrode handles, and (4) a desktop computer to control the apparatus. Figure 3 is a photo of our setup. We use an Agilent arbitrary waveform generator as the source of the

Figure 3. Proof-of-concept measurement setup. The test subject holds two brass electrode handles and the pulse signal is generated by an Agilent 33220A (20 MHz) arbitrary waveform generator. The receiver is an Agilent DSO3062A (60 MHz), 1 GSa/s digital storage oscilloscope.



pulse signal. Flexibility of the waveform generator is useful during the initial design phase and allows us to generate the required pulse waveforms in the final classifier. To measure the pulse waveform after the signal passes through a test subject we used an Agilent digital storage oscilloscope which allows storage of the waveform data for later analysis. The output of the waveform generator is connected to a brass handle that the user holds in the left hand. The other brass handle is connected to the oscilloscope signal input terminal. When a test subject holds one electrode in each hand the signal travels from the generator through the body and into the oscilloscope. To ensure exact triggering, the oscilloscope is connected to the synchronization output of the waveform generator.

7.2. Ethics and user safety

Our experimental prototype setup and its safety and methodology have been reviewed and authorized by the Central University Research Ethics Committee of the University of Oxford, under approval reference MSD-IDREC-C1-2014-156.

7.3. Biometric capture procedure

Each subject followed a specific procedure during the biometric measurement process to ensure that only minimal noise is introduced into the measured data. In the initial design phase, each test subject was sampled 10 times for each of the different signal types, for each voltage level and for various frequencies. Once we selected the pulse signal with the best results, samples were acquired for two data sets. The first consisted of 22 samples for each subject, taken in one measuring sessions, that is, at one point in time. The second included 25 samples per test person, obtained in five different sessions, over time. This was done to assess stability of the biometric over time.

The subject population included both males and females between the ages of 24 and 38. We sampled all test subjects at different times during the day over the course of several weeks. We tried to sample subjects in order to end up with sampling conditions as diverse as possible, for each subject. The interval between measurement sessions for the same

subject was varied between several hours and several weeks. This was done in order to try to eliminate any effects of sampling at a specific time of the day.

Data extracted from the measurement setup is in the form of a 4000 sample time-series describing voltage variation as seen by the oscilloscope. Figure 4 shows the input pulse sent by the waveform generator and the pulse measured by the oscilloscope.

Time series measurements are converted to the frequency domain using the FFT and the first 100 frequency bins of the FFT data are used for classification. Operating in the frequency domain has several advantages. First, there is no need to worry about alignment of the measured data pulses when computing metrics, such as the Euclidean distance between pulses. Second, it quickly became apparent that only lower frequency bins carry any distinguishing power. Higher frequency bins were mainly noise, meaning that the FFT can be used to perform dimensionality reduction of the original 4000 sample time-series to the vector of 100 FFT bins.

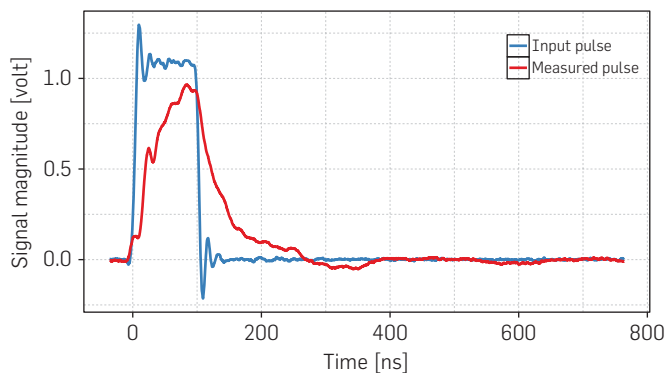
7.4. Results

We present two different classifiers: one for authentication and one for identification. The former is based on SVM and verifies a 1 : 1 match between a sample from an unknown person and that of a requested person. The identification classifier, also based on SVM, verifies a 1 : n match between a sample of a known person against all samples in a database. The identification classifier is of a closed-set variety. Section 2 provides a more detailed description of open- and closed-set classifiers.

We sub-divide results into: (1) those from a single test-set, which show the distinguishing power of pulse-response, and (2) those based on data sampled over time, which assess stability (permanence) of pulse-response.

Authentication classifier. Figure 5 shows the distinguishing potential of the authentication classifier applied to a data set collected over several weeks. Each bar shows the classifier’s performance for different threshold levels, for each of the test subjects. The threshold is a measure of assurance of correct identification. If a low false positive rate is acceptable, better sensitivity can be achieved. The classifier’s performance is measured using fivefold cross-validation to ensure statistical robustness. The figure shows that all subjects are

Figure 4. Input and output waveforms. One measurement consists of 4000 samples with the rate of 500 MSA/s.



recognized with a very high probability, as the true positive rate confirms.

Applying the authentication classifier to the single-session data set yields even better performance figures (see the full version of this paper in Rasmussen et al.⁸). For example, 10% false positives allow us to achieve sensitivity of almost 100%.

Identification classifier. Identification is a multi-class classification problem. Our classifier consists of multiple SVMs and follows a one-against-one approach (aggregation by voting). Due to this increased complexity a slight drop in performance is expected, in comparison to authentication, which is a binary classification task.

Results obtained from the identification classifier over the two data sets are shown in Figure 6. Even with increased complexity, the identification classifier performs very well on both data sets. The single-session data set contains ten people and the goal of the classifier is to identify each person as accurately as possible. There is a slight decrease in performance for the data set containing samples taken several weeks apart. The reason for this decrease is that samples taken far apart are influenced by very different conditions.

Figure 5. True positive rate for each test subject for the authentication classifier fed with the data sampled over time. Error bars show 95% confidence interval. The x-axis reflects the discrimination threshold for assigning the classifier’s prediction output to a positive or a negative.

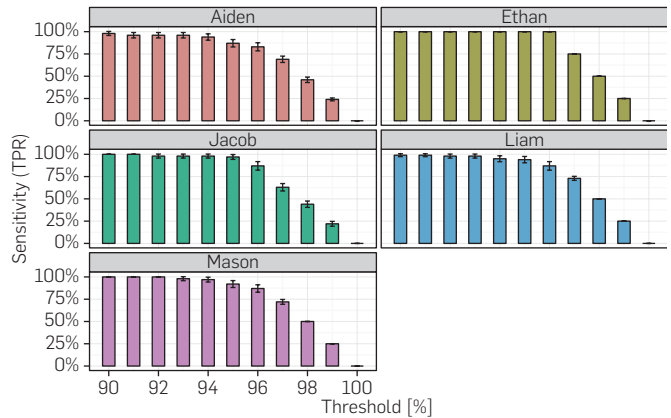


Figure 6. Identification classifier results. The true positive rate for each test subject is obtained by applying five times stratified fivefold cross-validation. Error bars show 95% confidence interval.

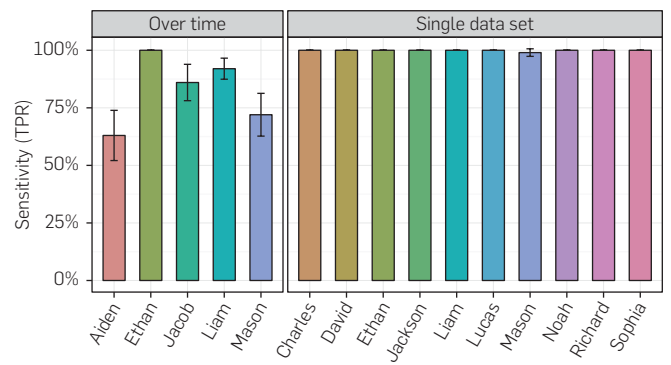


Table 1. Summary of results for authentication and identification classifiers, averaged over all users.

	TP	FP	TN	FN	In [%]		
					Sensitivity	Specificity	Accuracy
Authentication							
—Single set	2.0	0.0	18.0	0.0	100	100	100
—Over time	4.4	2.4	17.6	0.6	88	88	88
Identification							
—Single set	2.0	0.0	18.0	0.0	100	100	100
—Over time	3.4	1.6	18.4	1.6	68	92	87.2

All performance figures have been assessed on the basis of test data not involved in any development or training phase of the classifiers. Values for true/false positives/negatives are at the equal error rate of $EER = 0.00$ for the single data set and $EER = 1.12$ over time.

There might be physiological changes, such as weight loss or gain, or there might be differences in the ambient temperature, humidity, clothing, and a number of other factors.

Table 1 summarizes results for the two classifiers. Both classifiers can be tuned by selecting a specific false positive rate. For example, in a continuous authentication application, where false negatives are of greater concern, classifiers can be tuned to a lower false negative rate, by accepting a higher false positive rate.

8. RELATED WORK

The full version of this paper has a detailed survey of related work.⁸ In this version, we provide a brief overview.

Biometrics, as a means of recognizing an individual using physiological or behavioral traits, has been an active research area for many years. A comprehensive survey of conventional physiological biometrics can be found in Jain et al.⁵ While physiological biometrics tend to be relatively stable over time, they are sensitive to deception attacks, for example, mock fingers.¹ In contrast, behavioral biometrics are much harder to circumvent. However, the performance of behavioral biometric systems is usually worse and can require re-calibration due to normal variations in human behavior. Initial results on behavioral biometrics were focused on typing and mouse movements, for example, Spillane.⁹ Keystroke dynamics became quite popular,⁶ as a means to augment password authentication in manner similar to our PIN-entry scenario.

The result most closely related to our work is Cornelius et al.,² where bioimpedance is used as a biometric: a wearable wrist sensor passively recognizes its wearers based on the body's unique response to the alternating current of different frequencies. Experiments in Cornelius et al.² were conducted in a family-sized setting and show a recognition rate of 90% when measurements are augmented with hand geometry. The pulse-response biometric proposed in this paper solves a different problem but it also uses the body's response to a signal. It achieves a recognition rate of 100% when samples are taken in one session and 88% when samples are taken weeks apart (no augmentation is required in both cases).

9. CONCLUSION

We proposed a new biometric based on the human body's response to an electric square pulse signal. This biometric

can serve an additional authentication mechanism in a PIN entry system, enhancing security of PIN entry with minimal extra user burden. The same biometric is applicable to continuous authentication. To this end, we designed a continuous authentication mechanism on a secure terminal, which ensures user continuity, that is, the user who started the session is the same one who is physically at the terminal keyboard throughout the session.

Through experiments with a proof-of-concept prototype we demonstrated that each human body exhibits a unique response to a signal pulse applied at the palm of one hand, and measured at the palm of the other. Using the prototype we could identify users—with high probability—in a matter of seconds. This identification mechanism integrates well with other established methods, for example, PIN entry, to produce a reliable added security layer, either on a continuous basis or at login time. □

References

- Barral, C., Tria, A. Fake fingers in fingerprint recognition: Glycerin supersedes gelatin. In *Formal to Practical Security*. V. Cortier, C. Kirchner, M. Okada, and H. Sakurada, eds. Volume 5458 of *Lecture Notes in Computer Science* (2009). Springer, Berlin, Heidelberg, 57–69.
- Cornelius, C., Sorber, J., Peterson, R., Skinner, J., Halter, R., Kotz, D. Who wears me? Bioimpedance as a passive biometric. In *Proceedings of the USENIX Workshop on Health Security and Privacy* (August 2012).
- Information Technology Laboratory – National Institute of Standards and Technology. The biometrics resource center, 2013.
- Jain, A., Ross, A., Nandakumar, K. *Introduction to Biometrics*. SpringerLink, Bücher. Springer, 2011.
- Jain, A., Ross, A., Pankanti, S. Biometrics: A tool for information security. *IEEE Transactions on Information Forensics and Security* 1, 2 (June 2006), 125–143.
- Monrose, F., Reiter, M.K., Wetzels, S. Password hardening based on keystroke dynamics. In *Proceedings of the 6th ACM Conference on Computer and Communications Security, CCS'99* (New York, NY, USA, 1999). ACM, 73–82.
- National Science & Technology Council. Biometrics frequently asked questions, 2006.
- Rasmussen, K.B., Roeschlin, M., Martinovic, I., Tsudik, G. Authentication using pulse-response biometrics. In *The Network and Distributed System Security Symposium (NDSS)*. Volume 2 (2014).
- Spillane, R. Keyboard apparatus for personal identification. *IBM Technical Disclosure Bulletin* 17, 3346 (1975).
- Wikipedia. Sensitivity and specificity, 2013.
- Woodward, J., Orlans, N., Higgins, P. *Biometrics*. RSA Press Series. McGraw-Hill/Osborne, 2003.

Ivan Martinovic, Kasper Rasmussen, and Marc Roeschlin (ivan.martinovic,kasper.rasmussen,marc.roeschlin}@cs.ox.ac.uk) Department of Computer Science, University of Oxford, Oxford, United Kingdom.

Gene Tsudik (gene.tsudik@uci.edu), Department of Computer Science, University of California, Irvine, Irvine, CA.

CAREERS

Butler University

The Computer Science and Software Engineering Department at Butler University seeks a Postdoc Teaching Assistant. For details please see <https://www.butler.edu/hr/faculty-openings>. For inquiries please contact us at cs-se@butler.edu.

Connecticut College

Connecticut College Computer Science seeks a full-time tenure-track assistant professor with expertise in the areas of web technologies, mobile computing, and cybersecurity to join our department in August 2017. Please see <http://cs.conncoll.edu/job.html> for more details.

Indiana University School of Informatics and Computing

The Indiana University School of Informatics and Computing, Indianapolis, invites applications for a tenured associate or full professor in the growing field of data science or related area, to fill the position of Associate Dean for Research. Appointment begins August 1, 2017. Candidates must demonstrate an outstanding scholarly record of research, exhibited by high-impact peer-reviewed publications, a forward-looking, vigorous research agenda and a demonstrated history of securing significant, competitive external funding.

An exceptional researcher is sought to lead and expand the research enterprise of our school and contribute to the department's growing data science academic program. All areas of data science will be considered.

Qualifications

- ▶ Ph.D. in Computer Science, Information Science, Statistics, Data Science, or related discipline.
- ▶ Outstanding record of research productivity and impact.
- ▶ Record of significant external funding is required.
- ▶ Effective teaching for classroom, online, or blended learning.

Full position description and application instructions at <https://indiana.peopleadmin.com/postings/3224>.

Questions can be directed to Jeff Hostetler, Assistant to the Dean at jehostet@iupui.edu

The School of Informatics and Computing is eager to consider applications from women and minorities. Indiana University is an Affirmative Action/Equal Opportunity Employer. IUPUI is an Affirmative Action/Equal Opportunity Institution M/F/D/V.

Missouri University of Science and Technology

The Computer Science Department at Missouri S&T invites applications from dynamic and vision-

ary individuals for the position of Department Chair. The successful candidate will guide the department in directions that will further elevate its national and international stature as well as enhance the success of its students, faculty, and staff by achieving departmental and campus strategic visions. The successful candidate should demonstrate exceptional skills in recruiting and retaining a diverse group of faculty, promoting collaboration and superior written and oral communication with all stakeholders on and off campus, creating the conditions necessary for faculty and student development and creativity, acquiring campus resources essential for department growth, and seeking external resources for program enhancement through fund-raising efforts. A PhD in Computer Science, or a closely related area, with a demonstrated track record of scholarly accomplishments, effective teaching, and overall leadership is required. The Computer Science Department has a proud 50-year history of advancing the quality and breadth of its educational mission, and grants ABET-accredited BS, as well as MS and PhD degrees. Further details regarding this opportunity and departmental information may be found at: <http://cs.mst.edu/departmentchairsearch/>.

Interested candidates should electronically submit an application consisting of a cover letter, a curriculum vitae, a statement of leadership philosophy and research and teaching interests, and complete contact information for five references to the Missouri University of Science and Technology's Human Resources Office at: <http://hr.mst.edu/careers/academic/> (position # 67756). Application review will begin on January 15, 2017, and will continue until the position is filled. For more information prior to submitting an application, please contact the Search Committee Chair, Prof. Wayne Huebner, at: huebner@mst.edu.

Missouri S&T is an AA/EEO employer and does not discriminate on the basis of race, color, religion, national origin, sex, sexual orientation, gender identity, gender expression, age, disability or status as a protected veteran. Females, minorities, and persons with disabilities are encouraged to apply. The university participates in E-Verify. For more information on E-Verify, please contact DHS at: 1-888-464-4218.

Rutgers University

The Computer Science Department at Rutgers University invites applications for several tenure-track Assistant Professor positions focusing on (a) Data Science and AI, and (b) Computer Systems and Networking. Responsibilities include research, teaching undergraduate and graduate level courses in various fields of Computer Science and supervision of PhD students based on funded projects. The appointments will start September 2017.

Qualifications: Applicants should show evidence of exceptional research promise with potential for external funding, and commitment to qual-

ity advising and teaching. Hired candidates must complete their Ph.D. in Computer Science or a closely related field by August 31, 2017. Applications received by January 13, 2017 will be given priority.

To apply for the Data Science and AI positions, go to: apply.interfolio.com/39330.

To apply for the Computer Systems and Networking position, go to: apply.interfolio.com/39389. If you have further questions, please email the hiring committee: kagarwal@cs.rutgers.edu.

Swarthmore College

The Computer Science Department invites applications for one tenure-track position and multiple visiting positions at the rank of Assistant Professor to begin Fall semester 2017.

Swarthmore College is a small, selective, liberal arts college located 10 miles outside of Philadelphia. The Computer Science Department offers majors and minors at the undergraduate level.

Swarthmore College has a strong institutional commitment to excellence through diversity and inclusivity in its educational program and employment practices. The College actively seeks and welcomes applications from candidates with exceptional qualifications, particularly those with demonstrated commitments to a more inclusive society and world. For more information on Faculty Diversity and Excellence at Swarthmore, see <http://www.swarthmore.edu/faculty-diversity-excellence/information-candidates-new-faculty>

Applicants must have teaching experience and should be comfortable teaching a wide range of courses at the introductory and intermediate level. Candidates should additionally have a strong commitment to involving undergraduates in their research. A Ph.D. in Computer Science or near the time of appointment is required.

For the tenure-track position, we are interested in applicants whose areas fit broadly into theory and algorithms, systems, or programming languages. Priority will be given to complete applications received by November 15, 2016.

For the visiting position, strong applicants in any area will be considered. Priority will be given to complete applications received by February 1, 2017.

Applications for both positions will continue to be accepted after these dates until the positions are filled.

Applications should include a cover letter, vita, teaching statement, research statement, and three letters of reference, at least one (preferably two) of which should speak to the candidate's teaching ability. In your cover letter, please briefly describe your current research agenda; what would be attractive to you about teaching in a liberal arts college environment; and what background, experience, or interests are likely to make you a strong teacher of a diverse group of Swarthmore College students.

Tenure-track applications are being accepted online at



WISCONSIN
UNIVERSITY OF WISCONSIN-MADISON

David J. DeWitt Chair in Computer Sciences
UNIVERSITY OF WISCONSIN-MADISON

The Department of Computer Sciences is pleased to announce the establishment of the David J. DeWitt Chair in Computer Sciences, endowed by the generosity of alumnus Dr. Rakesh Agrawal. **Applications and nominations are currently being accepted.**

Scholars with an established track record in the area of database systems are encouraged to apply. Appointment will be as full professor with tenure. While we are especially interested in candidates with a strong "systems" focus, any scholar whose primary publication venues have been ACM SIGMOD, VLDB, ACM PODS and ACM SIGKDD is encouraged to apply.

The DeWitt Chair shall be an internationally renowned researcher and educator committed to benefitting society through technological innovation and the democratization of knowledge acquisition and diffusion. The recipient will be a key member of UW-Madison's database research group and join a highly ranked CS department with over \$20M of research activity per year, located in a vibrant city that Forbes named one of the fastest-growing tech hubs in the U.S.

Additional information can be viewed at
<http://www.cs.wisc.edu/about/employment>

Renaissance Technologies,

a quantitative financial management company trading in global financial markets, has openings for researchers and programmers at our Long Island, New York, research campus.

The ideal research candidate will have

- a PhD in computer science, mathematics, physics, statistics, or a related discipline
- a demonstrated capacity to do first-class scientific research
- computer programming skills

The ideal programming candidate will have

- strong analytical and programming skills
- an in-depth knowledge of software development in a C++ Unix environment

Experience in finance is not required.

"Renaissance is . . . the pinnacle of quant investing. No one else is even close."

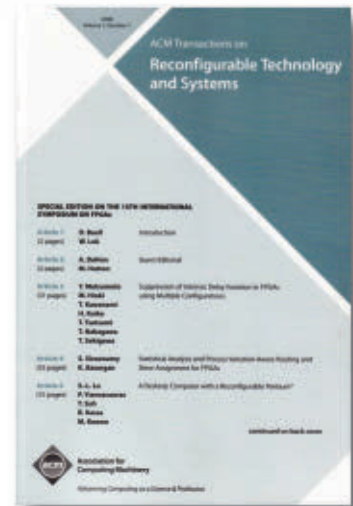
– Bloomberg Markets article, November 21, 2016

To apply, send your resume to careers@rentec.com.

For more information, visit www.rentec.com/careers.

Renaissance

ACM
Transactions on
Reconfigurable
Technology and
Systems



This quarterly publication is a peer-reviewed and archival journal that covers reconfigurable technology, systems, and applications on reconfigurable computers. Topics include all levels of reconfigurable system abstractions and all aspects of reconfigurable technology including platforms, programming environments and application successes.

www.acm.org/trets
www.acm.org/subscribe



Association for
Computing Machinery

Tenure-track faculty in Security and Privacy



The College of Information Sciences and Technology (IST) at The Pennsylvania State University invites applications for multiple tenured and tenure-track faculty positions in Security and Privacy. We seek exceptional candidates with a high quality research and publication record to strengthen and complement our existing research strengths. We welcome applications from scholars at all ranks and with a variety of disciplinary backgrounds including computer and information sciences, social and behavioral sciences, and engineering. We are seeking applicants from all areas of security and privacy, including emerging systems and applications (e.g., secure Internet of Things, secure mobile health), human factors, and the security and privacy aspects of Data Science (e.g., adversarial machine learning). Successful candidates will be expected to develop an internationally competitive, externally funded research program, and contribute to graduate and undergraduate education and training. The College of IST has a strong, externally funded research program in Security and Privacy. Current research strengths include systems and software security, privacy, human factors (e.g., usability), game and control-theoretic analysis, and several cross-cutting areas. Our College has other world-class research groups in Human Centered Design and in Data Science, and we are eager to strengthen our collaborations with these groups. Additional opportunities for research collaboration are offered by the Institute for Cyberscience, the Social Sciences Research Institute, and the Huck Institutes of the Life Sciences which serve as focal points for transformative research in areas of critical national priority. The College has a strong graduate program (with over 100 Ph.D. students), and a highly successful undergraduate program.

TO APPLY, visit <http://apptrkr.com/916192> please upload only basic personal information. In addition, interested applicants should submit the following material to <https://academicjobsonline.org/ajo/jobs/7725>: (1) a cover letter, (2) a Curriculum Vitae, (3) 3-5 page research statement, (4) a one-page teaching statement, (5) contact information of 3-5 professional references. Review of applications will begin on October 1, 2016, and will continue until the position is filled. The Pennsylvania State University is an equal opportunity employer and is committed to increasing the diversity of its faculty. Inquiries about the positions may be directed to Dr. Peng Liu, Faculty Search Committee Chair, College of IST, The Pennsylvania State University, University Park, PA 16802 or via email to facultyrecruiting@ist.psu.edu. The Pennsylvania State University is the land grant institution of Pennsylvania. University Park is the largest of Penn State's 24 campuses, with undergraduate enrollment of approximately 44,000 students and offering more than 150 programs of graduate study. The College of IST has award-winning faculty and state-of-the-art facilities. Both faculty and students are dedicated to collaboration and applying knowledge to make our lives better. University Park is located in Pennsylvania and is located in State College PA, ranked the 3rd safest metropolitan area in the United States by CQ Press, and the 8th best college towns in the nation by Best College Reviews.

CAMPUS SECURITY CRIME STATISTICS: For more about safety at Penn State, and to review the Annual Security Report which contains information about crime statistics and other safety and security matters, please go to <http://www.police.psu.edu/clery/>, which will also provide you with detail on how to request a hard copy of the Annual Security Report.

Penn State is an equal opportunity, affirmative action employer, and is committed to providing employment opportunities to all qualified applicants without regard to race, color, religion, age, sex, sexual orientation, gender identity, national origin, disability or protected veteran status.

Job URL: <http://apptrkr.com/916192>

<https://academicjobsonline.org/ajo/jobs/8018>

Visiting applications are being accepted online at

<https://academicjobsonline.org/ajo/jobs/8020>

Candidates may apply for both positions.

University of Oregon University of Oregon Department of Computer and Information Science Faculty Position

The Department of Computer and Information Science (CIS) seeks applications for two tenure track faculty positions at the rank of Assistant Professor, beginning September 2017. The University of Oregon is an AAU research university located in Eugene, two hours south of Portland, and within one hour's drive of both the Pacific Ocean and the snow-capped Cascade Mountains.

The open faculty positions are targeted towards the following three research areas: 1) high performance computing, 2) networking and distributed systems and 3) data sciences. We are particularly interested in applicants whose research addresses security and privacy issues in these sub-disciplines; additionally, we are interested in applicants whose research complements existing strengths in the department, so as to support interdisciplinary research efforts. Applicants must have a Ph.D. in computer science or closely related field, a demonstrated record of excellence in research, and a strong commitment to teaching. A successful candidate will be expected to conduct a vigorous research program and to teach at both the undergraduate and graduate levels.

We offer a stimulating, friendly environment for collaborative research both within the department, which expects to grow substantially in the next few years, and with other departments on campus. The department hosts two research centers, the Center for Cyber Security and Privacy and the NeuroInformatics Center. Successful candidates will have access to a new high-performance computing facility that opens in October 2016. The CIS Department is part of the College of Arts and Sciences and is housed within the Lorry Lokey Science Complex. The department offers B.S., M.S. and Ph.D. degrees. More information about the department, its programs and faculty can be found at <http://www.cs.uoregon.edu>.

Applications will be accepted electronically through the department's web site.

Application information can be found at <http://www.cs.uoregon.edu/Employment/>.

Applications received by December 15, 2016 will receive full consideration. Review of applications will continue until the positions are filled. Please address any questions to faculty.search@cs.uoregon.edu.

The University of Oregon is an equal opportunity/affirmative action institution committed to cultural diversity and is compliant with the Americans with Disabilities Act.

The University encourages all qualified individuals to apply, and does not discriminate on the basis of any protected status, including veteran and disability status. The successful candidate will have the ability to work effectively with faculty, staff, and students from a variety of diverse backgrounds.

[CONTINUED FROM P. 120] As money rolled in, I hired hundreds of workers, added print on demand to produce paper copies, and published more than 1,000 novels, all adaptations of best sellers that had gone out of copyright.

In its most advanced version, the system could construct entire paragraphs describing a location. For example, if the final battle between the detective and the murderer took place in a public park, the system would steal text from the website of the one nearest the town the reader had selected. If the town was, say, Redding, Connecticut, that would be Putnam Memorial State Park. If it was, say, Redding, California, then Turtle Bay Exploration Park would be the denouement site. There were limitations, of course. The artificial intelligence was not advanced enough to raise the question of whether the Revolutionary War general after whom Putnam Park was named really was himself a murderer who executed young men on charges of spying on his army or deserting from it without holding proper trials. NOVELS did not make the reader's computer perform an ethical analysis of a story but merely personalized it superficially to entertain the reader.

Five years and one billion dollars into this spectacularly successful solo-owned business, Mycroft Christie contacted me via email with ominous news I at first refused to believe. He was only an underling at the Federal Bureau of Investigation's statistics bureau, responsible for compiling the Uniform Crime Reports the FBI publishes annually. Apparently, like me, he was an ambitious nerd who had stumbled across an especially promising idea. For a couple of years, the Report had shown a surprising rise in the homicide rate in small cities and towns, even as big cities like Chicago or Philadelphia showed no increase at all. He contacted several of the local police departments that had reported their first murder in a decade, collecting various bits of information. Most of the murders remained unsolved, but in two cases the murderer had been caught at the scene of the crime, with one of my novels in his possession.

Christie accused me of creating an information system that helped real murderers plan their crimes. The premise of every mystery novel is that only a singularly clever detective is smart enough to solve the crime. Why did the

British police repeatedly ask Sherlock Holmes for help solving murders? Well, yes, the police were stupid, but also the mysteries were challenging, and only a genius could solve them. In the absence of the fictional detective, one of my murder mysteries could be the script for a perfect crime in the painfully real world. In small communities, many people were angry enough at their neighbors or relatives to kill them but feared their guilt would quickly be discovered. So they entered their data into one of my interactive novels and used it as the instruction manual for their crime.

Feeling angry myself, and not thinking through the possible consequences, I entered "Mycroft Christie" into the memory register of NOVELS that held the default victim name. Most readers would enter the name of a personal enemy for the victim in the murder mystery, but, if they failed to do so, they would read about the circumstances of Mycroft's death, along with the fictional evidence on his virtual corpse. I was astonished a few days later when the Facebook newsfeed reported the real-death murder of Mycroft Christie, considered newsworthy only because he had become one of the crime statistics it was his job to assemble. I clearly needed to disappear before his FBI colleagues discovered our email exchange and accused me of the crime, so I used my wealth to create a new identity.

My limousine is waiting outside, to take me to the airport, as I contemplate the main menu of NOVELS, trying to decide which option should end this story. Do I select **Enter** to add this testimony to the dataset? Do I select **Exit** to leave everything as it is, for my employees to inherit and use as they judge best? Or do I select **Erase** to destroy the entire system and protect the public from any further novel murders? Ah, yes. Perhaps I do know how this story ends. The CIA may pay me handsomely to use the system to write spy novels. If not the CIA, then ... you? □

William Sims Bainbridge (wsbainbridge.com) is a sociologist who taught classes on crime and deviant behavior at respectable universities before morphing into a computer scientist, editing an encyclopedia of human-computer interaction, writing many books on things computational, from neural nets to virtual worlds to personality capture, then repenting and writing harmless fiction.

© 2017 ACM 0001-0782/17/02 \$15.00



The image shows the cover of the journal 'ACM Transactions on Reconfigurable Technology and Systems'. The cover features a blue and white geometric design with a purple diamond at the top right. The title is prominently displayed in blue and yellow. Below the title, there is a table of contents listing several articles with their authors and page numbers. The ACM logo is visible at the bottom left of the cover.

ACM
Transactions on
Reconfigurable
Technology and
Systems

ACM Transactions on
Reconfigurable Technology
and Systems

SPECIAL SECTION ON THE 15TH INTERNATIONAL
SYMPOSIUM ON FPGAs

Article 1 14 pages	R. Boud M. Loh	Resource Estimation
Article 2 14 pages	A. Shalunov M. Loh	Speed Estimation
Article 3 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 4 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 5 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 6 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 7 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 8 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 9 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 10 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 11 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 12 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 13 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 14 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 15 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 16 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 17 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 18 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 19 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs
Article 20 12 pages	T. Madsen M. Loh	Systematic Estimation of Resource Usage in FPGAs

Association for
Computing Machinery

www.acm.org/trets
www.acm.org/subscribe

Association for
Computing Machinery

From the intersection of computational science and technological speculation, with boundaries limited only by our ability to imagine what could be.

DOI:10.1145/3028783

William Sims Bainbridge

Future Tense Fatal Guidance

*In a series of interactive murder mysteries,
I might not have done it, but, then again, maybe I did.*

YOU WILL NEVER be able to find me, so don't even try. I do not think I murdered Mycroft Christie, and you will never prove I did. Yes, I was responsible for creating NOVELS, the New Online Virtually Excellent Literature System, and, to profit from it, I published personalized mysteries. But I never anticipated how it would be misused, with the loss of thousands of lives, and accept no responsibility for that unintended consequence.

My fundamental idea was not entirely new, but how I developed it was revolutionary. Interactive novels had existed since at least the 1970s, originally published as paper books, then online as trees of webpages. At the end of each scene, the reader would have a choice. Standing before the dead body of, say, a rich baron, do you accuse his butler or his wife? If the butler, does he have an alibi or not? If the wife, what was her motive? Some of the most recent role-playing computer games used the same crude method to add complexity to the stories, with the added feature that the player could select a name for the main character. My innovation was taking this method to its logical extreme.

My first experimental novel took a year to program but was a spectacular success. Based on *The House of the Seven Gables* by Nathaniel Hawthorne, it offered the reader an electronic book with a beautiful tapestry of personalizations. To begin with, the software would replace the last name Pyncheon, which appears 397 times in the original 1851 novel, with the reader's own family name. The reader would then select first names, replacing Hepzibah, Phoebe, Alice, Jaffrey, and Colonel

Pyncheon. "New England" appears 20 times in the book and could become "New York," "the Carolinas," "Scotland," or whatever region the reader called home. "Massachusetts" appears only twice, but still the reader was given the opportunity to enter the name of a different district, as in, say, "Connecticut" or some county in England. Salem, the town in Massachusetts where the actual House of the Seven Gables still stands, is not mentioned at all. Indeed, the fact that the novel uses the word "town" 46 times without ever naming it gave me the Principle of Optional Location, applying to all the later projects. Other words could also be replaced, to bring the story into the world inhabited by the reader. For example, "As the Italian shouldered his hurdy-

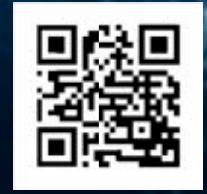
gurdy" could become "As the Texan shouldered his guitar."

The Principle of Optional Location was a starting point for my biggest innovation, a big data database with information stolen from Wikipedia, online phonebooks, and government sources that would automatically insert local landmarks into the novel, requiring the reader to merely specify the town. This worked especially well for murder mysteries, which tend to have formula plots and primarily vary their settings. Given the address of the place where the fictional murder was committed, NOVELS would insert the name of the nearest real restaurant where the detective could interview a possible witness to the crime, and perhaps the street address of the jail. [CONTINUED ON P. 119]



DEBS 2017

The 11th ACM International Conference
on Distributed and Event-Based Systems



Barcelona, Spain | June 19 - 23 | <http://www.debs2017.org>

The ACM International Conference on Distributed and Event-based Systems (DEBS) has become the premier venue for contributions in the fields of distributed and event-based systems. The objectives of the conference are to provide a forum dedicated to the dissemination of original research, the discussion of practical insights, and the reporting of experiences relevant to distributed systems and event-based computing. The conference aims at providing a forum for academia and industry to exchange ideas through industry papers and demo papers. The conference will also host a doctoral symposium, a workshop tutorials and a grand challenge competition. The winner of the grand challenge will be awarded a \$1000 cash prize also!

Important Dates

Abstract:	Feb 21, 2017
Research and Industry papers:	Feb 26, 2017
Tutorial proposal:	Mar 06, 2017
Grand Challenge:	Mar 29, 2017
Author Notification:	Apr 17, 2017
Poster, Demo, Doctoral Workshop:	Apr 29, 2017
Camera Ready Submissions:	Apr 19, 2017

General Chair:

Marta Patiño (Technical University of Madrid, Spain)

Research Track Co-Chairs:

Boris Koldehofe (TU Darmstadt, Germany)
Mani Chandy (Caltech, USA)

Industry Track Chair:

Mohammad Sadoohi (Purdue, USA)

Doctoral Symposium Co-Chairs

Buğra Gedik (Bilkent University, Turkey)
Leonardo Querzoni (Sapienza University, Italy)

Tutorial Co-Chairs:

Annika Hinze (University of Waikato, New Zealand)
Dave Eyers (University of Otago, New Zealand)

Grand Challenge Co-Chairs:

Vincenzo Gulisano (Chalmers University of Technology)
Zbigniew Jerzak (SAP Research)
Holger Ziekow (HS Furtwangen)



@ACM_DEBS

Computing Reviews

Connect with our Community of Reviewers

“I like CR because it covers the full spectrum of computing research, beyond the comfort zone of one’s specialty. I always look forward to the next Editor’s Pick to get a new perspective.”

- Alessandro Berni



Association for
Computing Machinery

ThinkLoud

www.computingreviews.com