Cryptocurrency Crime and Forensics ■ Malicious Browser Extensions ■ Cybersecurity Peer Instruction

IEEE

SECURITY & PRIVACY

Building Dependability, Reliability, and Trust

BLOCKCHAIN SECURITY AND PRIVACY

July/August 2018
Vol. 16, No. 4

Reliability Society

IEEE computer society

IEEE

IEEE SECURITY & PRIVACY

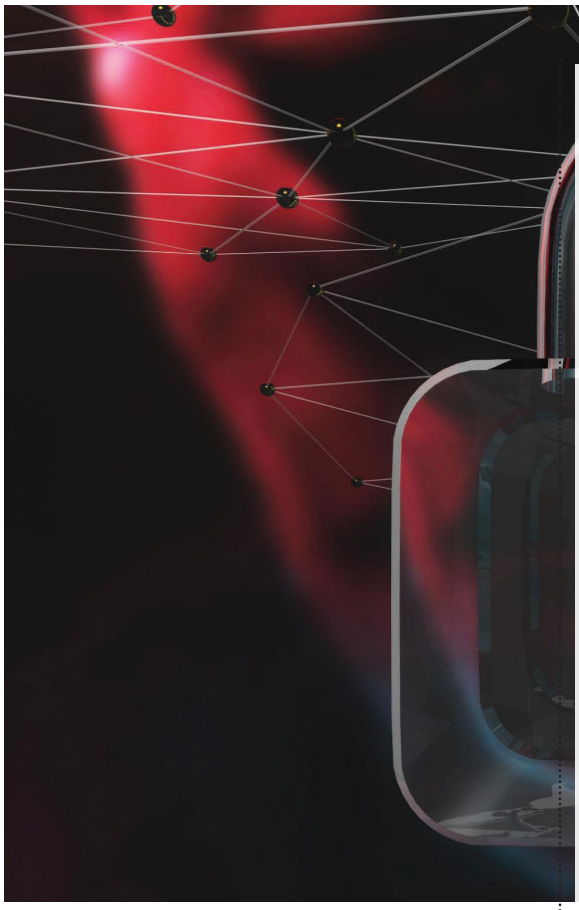
IEEE Security & Privacy magazine provides articles with both a practical and research bent by the top thinkers in the field.

- ✔ Stay current on the latest security tools and theories and gain invaluable practical and research knowledge,
- ✔ Learn more about the latest techniques and cutting-edge technology, and
- ✔ Discover case studies, tutorials, columns, and in-depth interviews and podcasts for the information security industry.

www.computer.org/subscribe

July/August 2018
Vol. 16, No. 4

Contents



Cover art by Barry Downard, www.debutart.com

Blockchain Security and Privacy

The blockchain emerged as a novel distributed consensus scheme that allows transactions, and any other data, to be securely stored and verified without the need of any centralized authority. Distributed trust and therefore security and privacy are at the core of the blockchain technologies, and have the potential to either make them a success or cause them to fail. This special issue of *IEEE Security & Privacy* is an attempt to collect the most interesting ideas from the community of researchers and professionals working on blockchain security and privacy.

11 Guest Editors' Introduction—Blockchain Security and Privacy
Ghassan Karame and Srdjan Capkun

13 Top Ten Obstacles along Distributed Ledgers' Path to Adoption
Sarah Meiklejohn

20 A First Look at Identity Management Schemes on the Blockchain
Paul Dunphy and Fabien A.P. Petitcolas

30 Tyranny of the Majority: On the (Im)possibility of Correctness of Smart Contracts
Lin Chen, Lei Xu, Zhimin Gao, Yang Lu, and Weidong Shi

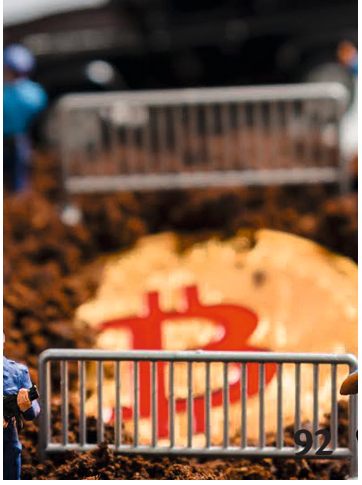
38 Blockchain Access Privacy: Challenges and Directions
Ryan Henry, Amir Herzberg, and Aniket Kate

46 When the "Crypto" in Cryptocurrencies Breaks: Bitcoin Security under Broken Primitives
Ilias Giechaskiel, Cas Cremers, and Kasper B. Rasmussen

57 PQChain: Strategic Design Decisions for Distributed Ledger Technologies against Future Threats
Rachid El Bansarkhani, Matthias Geihs, and Johannes Buchmann

Also in This Issue

66 Botnet in the Browser: Understanding Threats Caused by Malicious Browser Extensions
Raffaello Perrotta and Feng Hao



Columns

3 From the Editors
Encouraging Diversity in Security and Privacy Research
Terry Benzel

96 Last Word
You Are What You Eat
Daniel E. Geer Jr.

Departments

7 Interview
Silver Bullet Talks with Nick Weaver
Gary McGraw

82 Sociotechnical Security and Privacy
Inclusive Security and Privacy
Yang Wang

88 Education
Peer Instruction Teaching Methodology for Cybersecurity Education
Irfan Ahmed and Vassil Roussev

92 Cybercrime and Forensics
Cryptocurrencies—A Forensic Challenge or Opportunity for Law Enforcement? An INTERPOL Perspective
Giannis Tziakouris

Also in This Issue

87 | IEEE Reliability Society Information
95 | IEEE Computer Society Information

Postmaster: Send undelivered copies and address changes to *IEEE Security & Privacy*, Membership Processing Dept., IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854-4141. Periodicals postage rate paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8. Printed in the USA. **Circulation:** *IEEE Security & Privacy* (ISSN 1540-7993) is published bimonthly by the IEEE Computer Society. IEEE Headquarters, Three Park Ave., 17th Floor, New York, NY 10016-5997; IEEE Computer Society Publications Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720, phone +1 714 821 8380; IEEE Computer Society Headquarters, 2001 L St., Ste. 700, Washington, D.C. 20036. Subscribe to *IEEE Security & Privacy* by visiting www.computer.org/security. *IEEE Security & Privacy* is copublished by the IEEE Computer and Reliability Societies. For more information on computing topics, visit the IEEE Computer Society Digital Library at www.computer.org/csdl.

Encouraging Diversity in Security and Privacy Research

Terry Benzel
Associate Editor in Chief

Let's talk about diversity in security and privacy research. Over the past several months, I have been involved in several activities that focus on increasing participation by women and underrepresented minorities in the security and privacy research community.

First, I participated in the 2018 IEEE Security and Privacy conference. This is one of the top conferences for presenting research results in security and privacy. There, I gave a report on the GREPSEC series of workshops organized to encourage women and underrepresented minority graduate students to pursue research in security and privacy.

Then, I attended a presentation on broadening participation at the National Science Foundation (NSF). The NSF is taking a leadership role in addressing the serious problem of representation of women and underrepresented minorities in computer and information science and engineering (CISE) and across all scientific disciplines at NSF. This particular working group is focused on the CISE community.

Here, I give our readers a brief overview of these activities and remind them of the importance of proactive steps to increase diversity in the security and privacy community.

Participating in these discussions is both exciting and depressing. I am excited to see the number of people working to increase diversity and to learn about the variety of approaches that are being used to address this problem. At the same time, it is depressing to find myself still having these discussions after working for 35 years in such a lopsided community.

It feels entirely redundant to once again state the facts. It is well known that women and underrepresented minorities make up a small percentage of the computer security and privacy research community. While the number of women and minorities is low in computer science, hovering around 29 percent for masters degrees and 18 percent for doctorates, the situation in security and privacy research is considerably worse. There has been some increase in the number of women at the IEEE Security and Privacy conference, and women have taken on some leadership positions, but the field is still dominated by men. A recent study by Davide Balzarotti, "System Security Circus" (http://s3.eurecom.fr/~balzarot/notes/top4/index.html) reports that only 5 percent of the top 100 publications across four leading security conferences (ACM Computer and Communications Security [CCS], Usenix Security, IEEE Security & Privacy [Oakland], and Network and Distributed System Security [NDSS] Symposium) are written by women. At the same time, anecdotal evidence leads us to believe that women's participation in the top four security research conferences' program committees or as an author is under 10 percent. Nonetheless, attendance—not as an author or organizer—appears to be increasing, and women may represent as much as 20 percent. In addition, recent proactive efforts on the part of some conference organizers, for example, at the 2018 NDSS conference, increased female participation in the program committee to 14 percent. While some might point to these sorts of number as encouraging, they nonetheless affirm the very real problem.

GREPSEC: A Series of Workshops and a Growing Community

Starting in 2013, a small group of organizers and I led the formation of the GREPSEC workshop for women and underrepresented groups in security research. The workshop is now held biannually, with a one slated for 2019. They are designed to address one particular segment of the population and are targeted at graduate students in their second or third year. The workshops are kept to 30–40 students and 15–20 speakers/organizers and consist of research presentations by community

1540-7993/18/$33.00 © 2018 IEEE Copublished by the IEEE Computer and Reliability Societies July/August 2018

members, women, minorities, and others; "mentoring" is kept to a minimum. The goal of the workshop is to expose participants to the wide range of work being done in the field and to encourage one-on-one and small group interactions. Because the workshop is held the weekend before the IEEE Security and Privacy conference, many GREPSEC students also attend the conference. The workshops have been supported by the NSF, Computing Research Association, Google, Visa Research, and the IEEE Computer Society. An informal study indicates that since participating in the workshop:

- two GREPSEC attendees served on a major program committee,
- out of approximately 90, 57 had publications before or after attending GREPSEC,
- thirty-three had publications since GREPSEC,
- a total of 87 papers have been published by GREPSEC participants, and
- there have been 18 GREPSEC attendees' papers at major conferences.

This is statistically a small impact, yet in the years of the GREPSEC workshop, there is a noticeable difference at the conference in the number of women attendees and participants who are well integrated in the general conference.

Figure by womenscyberjutsu.org. Used with permission.

NSF CISE Broadening Participation

The second event that I had the opportunity to participate in was a presentation on the NSF CISE Broadening Participation in Computing effort. The NSF is a key funder of research in security and privacy and has a strong mission focused on education and supporting graduate students. NSF has provided direct support enabling students to travel and participate in conferences and provides significant support for workshops and conferences. NSF is committed to broadening participation in all of its activities and programs. This is reflected in the NSF Strategic Plan and subsequent effort such as NSF INCLUDES (https://www.nsf.gov/news/special_reports/nsfincludes/index.jsp) and the Committee on Equal Opportunities in Science and Engineering (CEOSE; www.nsf.gov/od/oia/activities/ceose/index.jsp).

The presentation and discussion that I participated in was specific to the CISE Directorate, home of the Secure and Trustworthy Cyber Space (SaTC) program. Efforts within CISE focus on addressing a broad class of underrepresented groups, including women, minorities (African Americans/Blacks, Hispanic Americans, American Indians, Alaska Natives, Native Hawaiians, Native Pacific Islanders, and persons from economically disadvantaged backgrounds), and persons with disabilities, in the computing field.

CISE has taken a number of steps to increase awareness around underrepresentation and to help members of the research community take constructive action to broaden participation at all levels. More information on the CISE Broadening Participation in Computing (BPC) effort is available at https://www.nsf.gov/cise/bpc and https://www.nsf.gov/pubs/2017/nsf17110/nsf17110.jsp.

CISE is in the process of implementing efforts to increase the community's engagement in BPC. This will include highlighting the effort in solicitations and requiring "meaningful" BPC activities in a growing set of programs. Specifically, under the pilot announced in NSF 17-110, frontier-class proposals to the SaTC and Cyber-Physical Systems programs were required to include meaningful BPC plans in 2017–2018. For proposals submitted in 2018–2019, all medium and large projects in CISE's core programs will be required to have approved BPC plans in place at the time of award. Along with these requirements, CISE will provide support for collective impact activities, training for NSF program officers and outside reviewers, and developing metrics for a range of activities spanning research, faculty development, university department development, and outreach activities. Inherent in the NSF steps is the recognition that broadening participation involves cultural change across many communities, ranging from formal and informal education institutions at the K–12 level to all aspects of college and university research and education.

The focus at NSF is encouraging as it has the ability to have an impact in both the short and long term due to its position as a major funder of research, development, education, and research dissemination. A long-term sustained commitment by NSF CISE with the goal of reaching parity is laudable and will contribute much to shifting the landscape.

Beyond Awareness

When confronted with this disparity, most organizations react with

campaigns aimed at increasing awareness, exposure, and outreach to the underserved communities as if believing that, with exposure to the opportunity, the problem would fix itself. However, a much harder problem is understanding and addressing the potential *causes* for the disparity. Increasing the exposure of underrepresented groups to opportunities in security and privacy is a potential solution if the problem is lack of exposure, but what other root causes are there, and what might be done about these? I encourage everyone to not only engage in outreach but to work to understand and change the underlying causes that have led to a smaller number of women and minorities in security and privacy research. The magazine editorial board seeks contributions exploring these questions.

As a woman working in security research, representation is always on my mind. In the past few months, I've seen the issue brought up in a number of forums. I am relieved to see serious efforts to increase awareness, help some limited portions of the community in the short to medium term, and work to create long-term sustaining focus on this problem. On the other hand, even with this increased focus and these numerous initiatives across the broad computer science and engineering communities, security and privacy research continues to have some of the lowest representation of women and underrepresented minorities.

There are many activities and initiatives targeted at different segments of the community along with all of the science, technology, engineering, and math (STEM) programs that deserve recognition. Obviously, scope and scale are key to success, as is addressing this issue at a societal scale and in the earliest days of education via K–12 and K–20 STEM efforts. For those interested in getting involved, there are resources available through professional societies, civic educational organizations, and private enterprises. Everyone reading this has a responsibility to actively work toward parity for underrepresented groups. ■

IEEE
SECURITY & PRIVACY
www.computer.org/security

EDITORIAL OFFICE
IEEE Security & Privacy
c/o IEEE Computer Society Publications Office
10662 Los Vaqueros Circle, Los Alamitos, CA 90720 USA
Phone | +1 714 821-8380; Fax | +1 714 821-4010

PUBLISHING COSPONSORS
IEEE computer society
IEEE Reliability Society

TECHNICAL COSPONSORS
IEEE Signal Processing Society
EMB IEEE Engineering in Medicine & Biology Society
IEEE CYBER SECURITY

Editor: Gary McGraw, gem@synopsys.com

Silver Bullet Talks with Nick Weaver

Gary McGraw | Synopsys

Hear the full podcast and find show links, notes, and an online discussion at www.synopsys.com/silverbullet.

Nicholas Weaver is a staff researcher at the University of California, Berkeley's International Computer Science Institute (ICSI). He also teaches courses at Berkeley. Weaver joined ICSI in 2003 as a post-doc after earning a PhD in computer science from Berkeley. His research focuses on network security, worms, botnets, and other Internet-scale attacks. He also works on network measurement.

ICSI is a nonprofit computer science research center. How is it funded?

It's almost entirely grant funded. As a researcher at ICSI, I'm very project and grant focused, and this is why I am doing more lecturing at Berkeley, because as a lecturer, I don't need to worry about research grants.

What are your views on ICSI tech transfer into the world?

As a research lab, we like building things that work. For example, the Bro Network Security Monitor was developed at ICSI, and that's being commercialized right now. Ten years ago, there was the extensible open router project, and there was a significant attempt to tech transfer that.

There are also systems that we've ended up building that have monetization models that don't match industry, but are productized. The Netalyzr network analysis tool that we originally wrote in Java in the web browser now runs on Android phones. We keep that running because it pays us in research results. We are able to turn the service into publications, and therefore we have a monetization strategy. It couldn't actually work out in the real world, but works for us. And we end up supporting a large number of users that way.

That's good stuff. You and I seem to share the same skeptical stance when it comes to cryptocurrencies and blockchain. Can you briefly give us a synopsis of your recent Burn It with Fire webinar?

I've come to this after five-plus years of watching the field and occasionally publishing on it. What it comes down to is there's actually three totally separate concepts. There is the concept of the cryptocurrencies themselves. There is the concept of the public blockchains, and then there is the concept of the private or permissions blockchains. Now let's start with the latter.

What is a private or permissions blockchain? Simply an append-only data structure with a limited number of authorized writers: aka, a git archive. There is nothing fundamental in a private blockchain that hasn't been understood in the field for 20-plus years. It's just it has a buzzword that causes idiots to throw money at the problem. If you see a private or permissions blockchain project, it means either one of two things. Either it's a delusional piece of techno-utopianism, or somebody smart in IT knows that there are real problems with what data you store, or how you access it, data provenance, and all this other stuff, and has bandied around this buzzword because idiots up in management will now throw money at this person to solve the real, interesting, hard problem.

That's one of the three. What about the other two?

The public blockchains are a global data structure where the idea is there is no centralized point of trust, but anybody can append to it. Now these systems are, let's say, not actually distributed as advertised. The Bitcoin blockchain is actually effectively controlled by only three entities, but in an attempt to be distributed, there is this religious notion that distributed trust is somehow good in and of itself. The result is systems that are either grossly inefficient or insecure.

The biggest tool that's used for these systems is what is called "proof

About Nick Weaver

Nick Weaver is a staff researcher at Berkeley's International Computer Science Institute (ICSI). He also teaches courses at Berkeley. Weaver joined ICSI in 2003 as a post-doc after earning a PhD in computer science from the University of California, Berkeley. His research focuses on network security, worms, botnets, and other Internet-scale attacks. He also works on network measurement. Weaver holds a BA in astrophysics and computer science. His thesis work was on FPGA architectures, but he's focused on computer security since 2001. He lives in Berkeley.

of work." And proof of work is best described as "proof of waste." The idea is that for somebody to rewrite the history, they have to do as much useless work as was done to create the history in the first place. Now this is great if you do a lot of useless work, except then it's inefficient. If you make the system efficient so you do not do a lot of useless work, you run into the problem of not actually having any real protection.

For example, Bitcoin, since the proof of work is paid for by the newly minted coins, ends up using as much power as New York City. It's just an obscene waste of energy. At the same time, these distributed public append-only ledgers only have been useful for cryptocurrencies. Now it's time to address the elephant in the room; the notion of the cryptocurrency itself.

Right? Back to one. Here we go.

Cryptocurrencies don't actually work as currency. They are provably inferior and can never be superior to the alternatives for real-world payments, unless you need what is known as "censorship resistance." If I want to transfer you $500 by PayPal, or Venmo, or whatever, we have these trusted intermediaries called banks, and they make it relatively cheap. However, there is a problem. If I want to transfer $500 to you for drugs or the like, these central authorities don't like it.

The only way to do censorship-resistant transactions without a cryptocurrency is cash, and cash requires physical proximity and math. One million in US dollars weighs 10 kilograms. That's a considerable amount of stuff to be lugging around. What a cryptocurrency is, well, let's do a direct to peer-to-peer payment system so that there are no central intermediaries, but let's do it electronically. This has been used quite practically for drug dealers, extortionists, fake hitmen, and all sorts of things like that. But if I want to do any payment that one of the central authorities will process, the cryptocurrencies provably don't work.

Let's say I want to buy a couch from Overstock.com using bitcoins. I have to turn my dollars into bitcoins, because I don't want to keep it in bitcoins because the price is jumping up and down. That is expensive. Transfer the bitcoin. That is relatively cheap right now, but it's been upwards of $30 in the past. And then the recipient on the other side has to convert the bitcoins back into dollars. You have these two mandatory currency conversion steps for any real-world transaction, and even Overstock, the one public company that supposedly embraces cryptocurrency, only keeps a few hundreds of thousands of dollars' worth of cryptocurrency, with the rest converted to dollars.

Cryptocurrencies do not work for legitimate purchases if you don't believe in the cryptocurrency. But let us suppose you believe in the vision of the great Satoshi. Then you don't want to use cryptocurrencies either, because they're baked in with these monetary policies that are designed to be deflationary. The first rule of a deflationary currency is never spend your deflationary currency.

There is one aspect of cryptocurrency that I think people don't understand, and it is this notion of tethers. Can you talk about that for a second?

There is a way to make a cryptocurrency work. You have to have an entity that takes dollars and gives you crypto dollars at par, and vice versa, that will take the crypto dollars and return you dollars. This is called a "bank," and these are called "banknotes," and it's recreating the 18th-century banking system. This can work, but one of three things has to happen. One option is you have regulation and enforce money-laundering laws and everything else, in which case you have a system that ends up being no cheaper or no more expensive than Visa, or Venmo, or anything else. What is the point?

Option number two is you have what is known as a "wildcat bank." This is a bank that prints banknotes that are actually unbacked. And this is a term from 18th-century banking.

The third option is a Liberty Reserve where you actually do back up your reserves. You redeem your digital banknotes, but you don't follow the money-laundering laws, in which case you end up being a guest of the federal government for the next 15 to 20 years.

At the same time, the money that the average person had is tied up temporarily or forever when the Feds shut down the institution. Tether is a specific cryptocurrency that promises to be backed

by dollars; they promise that there is this 1:1 ratio where you give them dollars, they give you tethers, and vice versa. The problem is this is almost certainly a wildcat bank because they managed to produce some 2 billion tethers in the space of a few months, and they are tied to a Bitcoin exchange that is otherwise cut off from banking. It may have been the direct reason why the Bitcoin price shot up so much.

Or they could be facilitating criminal money laundering, in which case those behind tether are liable to be guests of the federal government. This is, however, what actually enables most of the Bitcoin exchanges. Very few of the cryptocurrency exchanges actually are connected to the US banking system. You have Coinbase. You have Gemini, and you have Kraken (which should actually be shut down for other reasons of criminal activity, but that's neither here nor there). As for the rest of the exchanges, you can't actually transfer money into and out of them. These are where the hundreds and hundreds of different cryptocurrencies are actually traded on.

Tether has become this de facto reserve currency. If you look at Bitcoin trading volume, most of it is actually on tether-denominated exchanges and is not actually being exchanged for dollars, but these notional cryptodollars that may or may not be backed up, may or may not be a criminal enterprise—the flow just seems to continue on. It's really actually surprised me that it's lasted this long.

Yeah, it really is absolutely stunning this stuff. Thanks. That was extremely helpful. I think a lot of people need to have their eyes opened on this stuff, and you're one of the main people doing that.

I feel I have an obligation to. I kept looking at the field, and in the recent run up, I came to the conclusion that it's no longer harm-limited to a small population of self-selected believers. It is spilling out into the regular public.

Fortunately, I think the cryptocurrency space can die with proper application of regulation because of how the regulations already are, but it's become important for me to advocate for the need to clean up the space in that cryptocurrencies don't provide benefit to society. They don't provide benefit to all of us who aren't interested in committing crimes, but they do enable these problems. I think it is important to speak out. Another thing is the amount of scams in the space is just incredible.

Effectively every initial coin offering these days should be called a scam, because it is an unregistered security and wouldn't even pass the laugh test on Shark Tank. And we have got these people hyping smart contracts. Most of the cryptocurrency community seems intent on speed-running 500 years of economic history for choosing their bad ideas, but smart contracts are actually a new bad idea. The idea behind a smart contract is that I write a program that is not really a smart contract, it's a finance bot, because if it's a contract, you have this exception-handling mechanism called a judge in the legal system.

If I can walk up to a smart contract, say "Give me all your money," and it does, is that even theft? Well, it would be theft in the real world because we believe in justifying things, and this exception-handling mechanism of the judge and jury and all that. Smart contracts are instead—let's take the idea of a contract that is standardized and written in a formal way, it's called "legalese," and instead, rewrite it in a language that is uglier than JavaScript and has all sorts of pitfalls for programmers, eliminate the exception-handling mechanism, and then require that the code be bug free.

Except it's not bug free.

Oh, it's so amusingly not bug free. I like to use three examples. The first is the DAO, the Decentralized Autonomous Organization. The idea is, let's create a self-voting mutual fund for how we can invest our cryptocurrency in other projects. Now that there's actually nothing to invest was neither here nor there, but around 10 percent of all Ethereum at the time ended up in this basically self-creating, self-perpetuating, not-quite-a-Ponzi Ponzi scheme.

This was all fine and good until somebody noticed there was a reentrancy bug that allowed them to say, "Hey DAO, I am an investor. Give me all my money." And in the process repeat the thing as, "Hey DAO, give me all my money." And because there was a transfer then update, and you could re-entrantly call this code, it basically sucked all the money out.

The problem is, well, the money that was stolen mostly belonged to the people who came up with Ethereum in the first place. They basically did a code release that changed it and undid history. Their notion that code is law and there is no central authorities and no way to undo things was revealed to be a transparent lie when it's their money on the line.

Exactly.

So that's number one. Number two is the Proof of Weak Hands explicit Ponzi Scheme. Version 1.0 collected several million bucks before one bug locked it up so nobody could transfer any more money into it, and another bug allowed somebody to steal all the money in it. I think they're up to 3.0 now, which has yet to have a fatal bug, but we'll see how long that lasts.

Finally there is the Parity multisig wallet. One of the problems of cryptocurrencies is you can't actually store your cryptocurrency on an Internet-connected computer because if somebody gets onto your computer,

they get your private key and steal all your money. We actually had this happen to us in the early days of Bitcoin, and if security researchers can't use Bitcoin on an Internet-connected computer, nobody can. The idea is, let's make it a two-party check system. We will have three private keys, and you have to use two of them to transfer the currency.

This gives you good controls if you can theoretically maintain at least two of your cryptographic keys. Some systems, like Bitcoin, offer it as a primitive. For Ethereum, it was built as a smart contract on top of things. This was the Parity multisig wallet, which collected some hundreds of millions of dollars, including an ICO by the guy behind the Parity multisig wallet. Until somebody noticed that there was a bug where you could go up to one of these wallets say, "Hey, wallet. You belong to me. Hey, wallet. Give me all your money," and started cleaning these out. And the only reason this wasn't a $150 million theft is somebody else noticed that this was going on, stole all the money first, and then gave it back to the victim once the victim had upgraded code.

Unbelievable.

Which gets better. Now there's the upgraded wallet code. For efficiency, everybody refers to the same wallet contract, and there was a bug in this contract. Some random loser came along and said, "Hey contract. You belong to me now," and the contract said, "Okey-doke. Yeah, I do." Okay, oh crap. This shouldn't have happened. "Hey, contract. Kill yourself." The contract committed suicide, and now $150 million worth of cryptocurrency is locked up and effectively inaccessible unless the central authorities, that aren't supposed to exist, change the code to unlock this. We're not done yet. The pièce de résistance.

The lead programmer and shining light behind this fiasco is the guy who invented the programming language in the first place. The problem is these things are designed to be non-upgradeable, but there are hacks that allow you to update them. If your money is tied up in somebody else's contract because their contract is the service, you have a choice. Either that contract has to have been bug free when created, not good, or that contract has to be upgradeable, in which case you have to trust that they upgrade the contract properly and don't cause damage or work against you in the process.

You have a central authority again.

You have a central authority. For example, there was a bug discovered in some of these smart contracts that run these ICOs, where somebody was able to create, what was it, 200 billion new tokens? Well, the people in charge of that particular smart contract were able to undo the process, but that means also if they can destroy the hack-created tokens, if you're invested in them, they can destroy your tokens too if they feel like it.

You have to trust them.

This is the ultimate irony in all these systems—their belief in this mantra that lack of trust and decentralization are good in and of themselves, ignoring the huge advantages you get with just even the slightest of smattering of centralized trust. Yet they end up building systems that aren't even decentralized. They build things that are orders of magnitude less efficient than they could be, but which have central authorities and aren't distributed anyway.

I think the real design decision was, "I would like to have all the trust belong to me."

No, the cryptocurrency community truly believes in this idea of decentralization; that you should have to trust nobody.

They're just bad at implementing it.

They don't understand the costs involved in that, and they cannot seem to ever implement it that way anyway.

All right, so onto a very personal issue. You suffer from depression that's treated by therapy and medication, and you talk about that so others can benefit from the good aspects of treatment and therapy. Tell us a little bit about that.

I've basically had in my life multiple depression meltdowns, and therapy and drugs saved my life twice as a student. And both times, after about a year, I'd just go off the medication, and a couple of years later the same thing would happen again. Just after the third incident, I realized that I didn't want to repeat that mistake.

So, when I'm teaching students, every semester I include in my first slide deck, the notion that yes, I've been there. I've done that. This is not good. There is help available. Every semester at least one student has proven that it's been worthwhile and they'll come up to me afterward.

Super important work. Last question, what is your favorite fiction book or your favorite fiction book you're reading at the moment?

Let's just say I'm a huge fan of *The Laundry Files*.

The Silver Bullet Podcast with Gary McGraw is cosponsored by Synopsys and this magazine and is syndicated by SearchSecurity. ■

Gary McGraw is vice president of security technology at Synopsys. He's the author of *Software Security: Building Security In* (Addison-Wesley 2006) and eight other books. McGraw received a BA in philosophy from the University of Virginia and a dual PhD in computer science and cognitive science from Indiana University. Contact him via garymcgraw.com.

Blockchain Security and Privacy

Ghassan Karame | NEC Laboratories Europe
Srdjan Capkun | ETH Zurich

The blockchain emerged as a novel distributed consensus scheme that allows transactions, and any other data, to be securely stored and verified without the need of any centralized authority. For some time, the notion of blockchain was tightly coupled with a now well-known proof-of-work hash-based mechanism of Bitcoin. Today, there are more than a hundred alternate blockchains: some are simple variants of Bitcoin, whereas others significantly differ in their design as well as provide different functional and security guarantees. This shows that the research community is in search of a simple, scalable, and deployable blockchain technology. Various reports further point to an increased interest in the use of blockchains across many applications and to a significant investment in the development of blockchains by different industries. It is expected that the blockchain will induce considerable change to a large number of systems and businesses.

Distributed trust and therefore security and privacy are at the core of the blockchain technologies, and have the potential to either make them a success or cause them to fail.

This special issue aims at collecting the most relevant ongoing research efforts in blockchain security and privacy. We are very grateful to this community, especially for its vivacity and vast participation.

The issue starts with an introductory article written by Sarah Meiklejohn, "Top Ten Obstacles along Distributed Ledgers' Path to Adoption," which presents hindrances preventing the widespread adoption of the blockchain technology by the community and outlines potential avenues of research.

In their article, "A First Look at Identity Management Schemes on the Blockchain," Paul Dunphy and Fabien A.P. Petitcolas discuss a number of identity management schemes based on the blockchain and evaluate three schemes—uPort, ShoCard, and Sovrin—using a novel framework.

In "Tyranny of the Majority: On the (Im)possibility of Correctness of Smart Contracts," Lin Chen and colleagues tackle the correctness of smart contracts in the blockchain. More

1540-7993/18/$33.00 © 2018 IEEE Copublished by the IEEE Computer and Reliability Societies

specifically, they analyze consensus of smart contract results in decentralized systems and show that the correct execution results of smart contracts are not always accepted as consensus.

In "Blockchain Access Privacy: Challenges and Directions," Ryan Henry and colleagues discuss another important problem in the blockchain: privacy. They show that Tor offers limited privacy and illustrate the need for research "beyond Tor" to tackle important access privacy issues in contemporary blockchains.

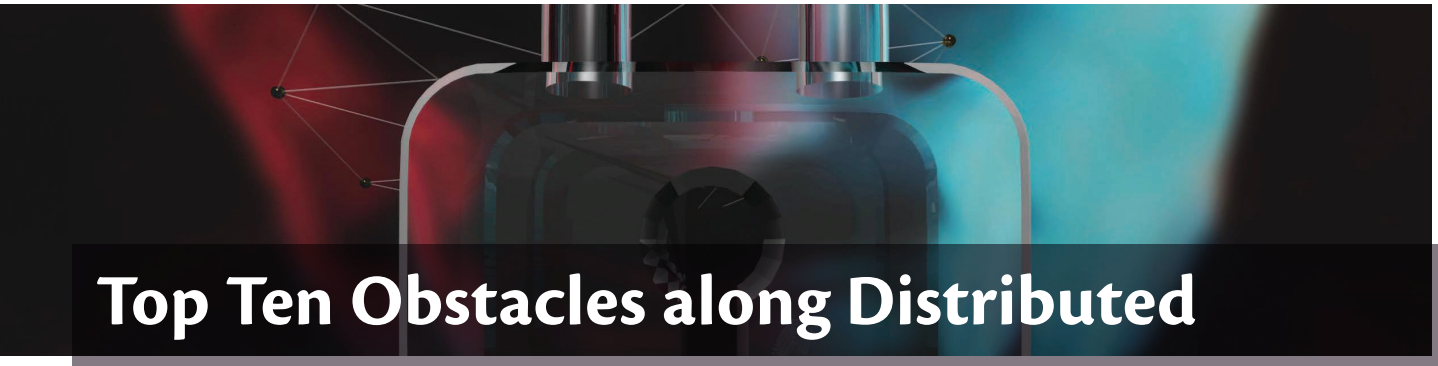
"When the 'Crypto' in Cryptocurrencies Breaks: Bitcoin Security under Broken Primitives," by Ilias Giechaskiel and colleagues, presents an analysis of the effect of broken primitives on Bitcoin. This analysis leads to several suggestions for the Bitcoin migration plans and insights for other cryptocurrencies in case of weakened cryptographic primitives.

Finally, Rachid El Bansarkhani and colleagues extend this analysis in their article "PQChain: Strategic Design Decisions for Distributed Ledger Technologies against Future Threats," suggesting paths for a secure instantiation of the blockchain protocol, taking into account the presence of large-scale quantum computers and potential future attacks against the underlying hash functions.

We hope you enjoy this special issue! ■

Ghassan Karame received his MS in information networking from Carnegie Mellon University (CMU) in December 2006 and his PhD in computer science from ETH Zurich, Switzerland, in 2011. Between 2011 and 2012, he worked as a postdoctoral researcher at the Institute of Information Security of ETH Zurich. Since 2012, Karame joined NEC Laboratories Europe where he is currently the manager and chief researcher of the Security group. Karame is interested in all aspects of security and privacy with a focus on cloud security, SDN/network security, and Bitcoin/blockchain security. He is a member of IEEE and ACM and is the author of several papers and patent applications. More information can be found at http://ghassankarame.com. Contact him at ghassan@karame.org.

Srdjan Capkun is a full professor in the Department of Computer Science, ETH Zurich, and director of the Zurich Information Security and Privacy Center (ZISC). He was born in Split, Croatia. He received his Dipl.Ing. degree in electrical engineering/computer science from the University of Split in 1998 and his PhD in communication systems from EPFL in 2004. Prior to joining ETH Zurich in 2006, he was a postdoctoral researcher in the Networked & Embedded Systems Laboratory (NESL), University of California, Los Angeles, and an assistant professor in the Informatics and Mathematical Modelling Department, Technical University of Denmark (DTU). His research interests are in system and network security. One of his main focus areas is wireless security. He is a co-founder of 3db Access, a spin-off focusing on secure proximity-based access control. Contact him at capkuns@inf.ethz.ch.



Top Ten Obstacles along Distributed Ledgers' Path to Adoption

Sarah Meiklejohn | University College London

This article presents the top ten obstacles toward the adoption of distributed ledgers, ranging from identifying the right ledger to use for the right use case to developing scalable consensus protocols that provide some meaningful notion of public verifiability.

In January 2009, Bitcoin was released into the world by its pseudonymous founder, Satoshi Nakamoto. In the ensuing years, this cryptocurrency and its underlying technology, called the *blockchain*, have gone on a rollercoaster ride that few could have predicted at the time of its deployment. It has been praised by governments around the world, and people have predicted that "the blockchain" will one day be like "the Internet." It has been banned by governments around the world, and people have declared it "adrift" and "dead." The price of Bitcoin skyrocketed in late 2013 up to $1,200 per bitcoin, only to spend the entire next year languishing at anywhere from $200 to $500 per bitcoin, before beginning a steady climb in 2016 that now has Bitcoin's price hovering around $4,500 per bitcoin (a number that will no doubt be wildly out of date at the time of publication).

After years in which discussions focused entirely on Bitcoin, people began to realize the more abstract potential of the blockchain, and "next-generation" platforms such as Ethereum (www.ethereum.org), Steem (http://steemit.com), and Zcash (http://z.cash) were launched. More established companies also realized the value in the more abstract properties of the blockchain—resilience, integrity, and so forth—and repurposed it for their particular industries to create an even wider class of technologies called *distributed ledgers* and to form industrial consortia such as R3 (www.r3.com) and Hyperledger (www.hyperledger.org). These more general distributed ledgers can look, to varying degrees, quite unlike blockchains and have a somewhat clearer (or at least different) path to adoption given their association with established partners in industry. As we describe below, however, they must nevertheless overcome many of the same obstacles to become truly productive and long-lasting solutions.

Amidst many unknowns, what is increasingly clear is that, even if they might not end up quite like the Internet, distributed ledgers—in one form or another—are here to stay. Nevertheless, a long path remains from where we are now to widespread adoption—that is, a point at which the blockchain is even within several orders of magnitude as ubiquitous as the industries it has the potential to disrupt, such as traditional databases or the Internet—and there are many important decisions to be made that will affect the security and usability of any final product.

In what follows, we present the top ten obstacles along this path as well as what we as a community can do (and have been doing) to address them. By necessity, many interesting aspects of distributed ledgers, both in

1540-7993/18/$33.00 © 2018 IEEE Copublished by the IEEE Computer and Reliability Societies

terms of problems and solutions, have been omitted, and the focus is largely technical in nature. Nevertheless, our hope is that this serves as an introduction to the topic and to potentially fruitful avenues of research.

10. Usability: Why Use Distributed Ledgers?

The problem. What does usability refer to, and how is it a problem? There are a few possibilities in this context; for example, one could say that currently deployed distributed ledgers do not have particularly nice interfaces and are difficult to use without expert knowledge. While this is certainly true, one can and should expect these problems with any new technology and further expect them to be resolved with time and increased usage. (Here relational databases and the development of graphical, if imperfect, user interfaces into them might serve as a useful history.)

Thus, a perhaps more interesting "usability" problem is: What do end users actually want from distributed ledgers? Most deployed cryptography that end users interact with today maps easily to processes and desires that have existed for centuries. If you want to hide the contents of your conversation with someone, use encryption. If you want to convince someone that it's really you saying something, use a (digital) signature. But what is it that the full public verifiability (or accountability, immutability, and so on) of distributed ledgers really maps to in terms of what end users want?

Potential and developing solutions. While some research has explored users' perceptions of Bitcoin,[1] there has been very little research into the broader usability of distributed ledgers or into the way they fit into existing mental models. It may be that in fact most end users have no interest in the properties of distributed ledgers (just as they also likely have no interest in the properties of regular databases). Given the growing number of users who already interact with platforms like Bitcoin and Ethereum, however, and the discussed potential for distributed ledgers to—among many other use cases—inform the choices of ethical consumers by providing supply chain transparency, it is still worth exploring which specific benefits people feel they can achieve by using these technologies.

9. Governance: Who Makes the Rules?

The problem. The beauty of distributed ledgers is that no one entity gets to control the decisions made by the network; in Bitcoin, for instance, coins are generated or transferred from one party to another only if a majority of the peers in the network agree on the validity of this action. While this process becomes threatened if any one peer becomes too powerful, there is a larger question looming over the operation of these decentralized networks: Who gets to decide which actions are valid in the first place? The truth is that all these networks operate according to a defined set of rules, and that "who makes the rules matters at least as much as who enforces them."[2]

In this process of making the rules, even the most decentralized networks turn out to be heavily centralized. In Bitcoin, the rate at which new blocks are added to the chain, the reward participants receive for sealing transactions into the ledger, and many other parameters were all handed down by Satoshi Nakamoto and have not changed since. The one parameter that users are proposing to change, the size of blocks, has led to a contentious debate that has been raging for years. In contrast, Ethereum has now carried out four so-called "hard forks" of its network, in which participants are essentially mandated to switch to a new version of the software to comply with new rules created by a core set of developers.

These increasingly common collapses in the governance structures underlying cryptocurrencies threaten to harm their value and reveal the issues associated with ad hoc forms of governance. Thus, the problem is not just that we don't know how to govern these technologies, or that the governance structures are both centralized and relatively fractured, but that—somewhat ironically—we need more transparency around how these structures operate and who is responsible for which aspects of governance, so that users of these cryptocurrencies can be aware of them and make informed decisions about their own level of involvement.

Potential and developing solutions. Several research papers have focused on ways to maintain decentralization in terms of the entities that enforce the rules,[3,4] in the form of a mining process that resists pooling, although in more general distributed ledgers, it is still very much an open question how to incentivize participants to help maintain the ledger. (For example, it is unclear who, other than Google or other large certificate authorities, would want to expend the resources necessary to maintain a log server in Certificate Transparency [www.certificate-transparency.org], which provides no explicit reward to these entities.)

In terms of the entities that make the rules, very few solutions have been proposed. The "Satoshi Oath" (http://ipfs.b9lab.com:8080/ipfs/QmXysWEAexXQqYZhTGpECvksnaBkSEWHdGhM7vNeHxu_e2g) outlines a pledge that the authors deem necessary for anyone setting out to create a blockchain-based application, but it still serves only as a guideline. Broader distributed ledger projects such as R3 and Hyperledger have opted for a consortium-based approach, and other projects such as Certificate Transparency have opted for a fairly centralized approach. As in Item 8, however, it is unclear how these approaches compare to one another or how the governance structures will evolve as these platforms gain popularity.

8. Meaningful Comparisons: Which Is Better?

The problem. Bitcoin was the first cryptocurrency to be based on the architecture we now refer to as the blockchain, but it certainly isn't the last; there are now thousands of alternative cryptocurrencies out there, each with its own unique selling point. Ethereum offers a more expressive scripting language and maintains state, Litecoin (http://litecoin.org) allows for faster block creation than Bitcoin, and each new ICO (http://en.wikipedia.org/wiki/Initial_coin_offering) promises a shiny feature of its own. Looking beyond blockchains, there are numerous proposals for cryptocurrencies based on consensus protocols other than proof of work (see Item 3) and proposals in non-currency-related settings, such as Certificate Transparency, R3 Corda, and Hyperledger Fabric, that still fit under the broad umbrella of distributed ledgers.

The natural issue that arises in such an increasingly crowded landscape is how to distinguish between these solutions and pick the one that is best for a given application. Do you need a blockchain or just a database? Maybe an Excel spreadsheet is sufficient for what you need to do? Related back to Item 10, what even are the properties that you want to satisfy? Do you need full public verifiability (and if so, why)? Even if someone could specify a list of necessary properties, it is still not clear which current platforms support which properties, and to what extent.

Potential and developing solutions. Some recent research[5] has looked at the different parameters chosen by different cryptocurrencies (for example, Bitcoin's 10-minute block generation versus the 2.5-minute interval used by Litecoin) and the effect these parameters have on the security of the system, finding, for example, that the same level of security against so-called "selfish mining" attacks is achieved by 37 blocks in Ethereum as by 6 blocks in Bitcoin (due to the relative stale block rates of the two platforms). While insightful, this work focuses specifically on cryptocurrencies based on proof of work and thus does not extend to more general distributed ledgers or cryptocurrencies based on alternative consensus protocols (see Item 3 for examples).

In considering distributed ledgers as a whole, recent research[6] explored the connections between the security properties provided by Certificate Transparency and Bitcoin, finding that the tradeoff between the two seemed to be between trust in a distributed set of authorities on the one hand and the need to (inefficiently) broadcast messages and engage in consensus protocols like proof of work on the other hand. This research again focuses on a specific underlying structure, so significant work remains to allow comparisons between other platforms, or more generally to understand the set of tradeoffs that one should consider in evaluating any particular choice.

7. Key Management: How to Transact?

The problem. Over the years, many Bitcoin users have reported incidents in which they have lost all their bitcoins because they threw away a hard drive, forgot a password, or didn't take appropriate measures to secure the wallets on their computers; in the worst publicly reported incident, a user lost 7,500 bitcoins. Indeed, once the hard drive containing the wallet is lost or your money is stolen, the irreversibility of both the underlying cryptography and the transactions means that there's no way to recover. In one very public example, an Ethereum smart contract called The DAO (short for Decentralized Autonomous Organization), designed to act as a collective investment vehicle, managed to attract over $150 million in what is considered the largest crowdfunding campaign ever (https://www.nytimes.com/2016/05/22/business/dealbook/crypto-ether-bitcoincurrency.html). After an attacker exploited a loophole in the code of the smart contract behind The DAO, the community was essentially powerless to do anything except watch them steal all the funds stored inside (until, that is, some of the governing Ethereum developers discussed in Item 9 created and advocated for a hard fork to undo the damage).

While these incidents obviously have serious financial implications for the individuals involved in them, they should in some sense not be particularly surprising to these individuals given the unregulated "Wild West" world of cryptocurrencies. People are proposing more mainstream uses of cryptocurrencies—for instance, integrating Bitcoin wallets into Linux distributions—that will put these wallets in the hands of less experienced users who may be less prepared for these types of risks; however, this will lead to wider and more serious consequences. We thus need robust solutions that can better tolerate both the loss and the theft of keys.

Potential and developing solutions. A commonly used Bitcoin technique that has the potential to mitigate this issue is a *multisignature*, in which multiple parties join their public keys together to create an address for which some subset of their signatures is sufficient to spend its contents. For example, in a 3-of-3 multisignature address, all three parties would need to contribute signatures in order to spend coins from the address, and in a 1-of-3 address, any one of the signatures would suffice. A 2-of-3 multisignature address arguably provides the best defense against key loss, as funds stored in the address are still accessible even if one key is lost, but an attacker would need to access two separate keys to steal from it.

One could similarly argue that secret sharing, in which the key is split among some number of friends (or

devices) who can be trusted to not collude but to provide their shares if and when key loss occurs, could also help. With both these solutions, however, we must understand if the threat model in which they work is realistic for the scenario (going back to Item 10) in which we expect to be distributing, storing, and using these keys. It is also still an open problem to identify solutions in which one could regain access to lost funds with the same ease with which users currently regain access to accounts for which they have forgotten the password.

6. Agility: Which Algorithms Do We Use?

The problem. As discussed in Item 9, in many cryptocurrencies, the rules seem to have been handed down from on high: for example, Bitcoin addresses are computed by taking the ECDSA public key, performing SHA-256, performing extended RIPEMD-160, performing SHA-256 again, and again, rearranging the bytes of this output and the output of the extended RIPEMD-160 hash, and converting the result into a base58 string. Similarly, once the governing developers have decided on a proof-of-stake protocol that they approve of (more on that in Item 3), it will replace Ethash as the consensus protocol in Ethereum, and everyone will have no choice but to agree if they wish to keep using the cryptocurrency.

Aside from the governance issues raised by these rigid specifications, cryptographic primitives break, and they do so frequently. Eventually, computers may become powerful enough that SHA-256 will be broken, which would allow anyone to rewrite the entire history of a blockchain secured using only hashes. While there would likely be sufficient warning before this happened to give the developers time to switch to more secure alternatives (and anyway the problem would go far beyond distributed ledgers!), there is an argument to be made for providing users with multiple options even today, just as is done with systems such as TLS. As with TLS, however, agility can enable dangerous attacks, so significant caution is needed in both how it is implemented and in which cryptographic primitives are supported. The arguably safer option is not to enable users to engage with multiple cryptographic primitives, but instead allow them to choose the consensus required for their transactions (see Item 3 for more on different consensus protocols), depending on their trust assumptions about both the people with whom they transact and the maintainers of the system.

Potential and developing solutions. Almost all cryptocurrencies achieve no agility whatsoever: for each cryptographic primitive, there is one valid instantiation, and there is only one way to achieve consensus. One arguable exception is Ethereum, in which one could technically encode any cryptographic primitive—even something that is completely insecure—into a smart contract by including a custom library within the contract code. Nevertheless, the underlying cryptography (that is, the checks that peers in the network make to verify individual transactions) and consensus protocol are just as rigid as in other cryptocurrencies.

Perhaps because they are unencumbered by the ideological mindset frequently encountered in cryptocurrencies, industry-led distributed ledgers are much more agile. In Corda, for example, an individual user can pick from a selection of available algorithms when forming a contract. In Hyperledger Fabric, participants can essentially plug in their own consensus protocol, leading to many different possible instantiations of the abstract design. Here then, the question is not necessarily whether or not it can be done at all, but how these different options can compose (for instance, what does it mean if two parts of the ledger have been agreed on by two distinct consensus protocols?) and what effect these options have on the overall security of the system.

5. Interoperability: How to Talk to Each Other?

The problem. Some people believe that the future will contain one single ledger (like the Internet), and in fact general-purpose platforms could in theory—that is, if the other issues on this list were solved—support most known applications of distributed ledgers. The more likely scenario, however, is that different companies will gravitate toward different ledgers based on their particular requirements. For example, financial applications requiring consensus between a fixed set of banks are more likely to adopt a platform such as Corda or Hyperledger Fabric, whereas applications that require fully open participation are more likely to adopt a platform such as Bitcoin or Ethereum.

To achieve the much-discussed potential of distributed ledgers to eliminate (or at least significantly open) the existing landscape of proprietary data silos, it is thus essential to achieve some kind of interoperability or a set of methods that allows these disparate ledgers to talk to each other.

Potential and developing solutions. Within the realm of platforms based on blockchains, the idea of a sidechain provides a simple way to translate actions from one blockchain into another—that is, to have transactions published in one ledger have an effect on another ledger. The security of these sidechains has been relatively unstudied to date, however, and while they have attracted significant attention, they have not yet seen much adoption. There are also several other methods proposed to transact across the blockchains of different cryptocurrencies, such as *atomic cross-chain trading* (http://en.bitcoin.it/wiki/Atomic_cross-chain_trading), but

again these solutions have been neither particularly well studied nor adopted in any meaningful way.

Beyond this, there have been a number of attempts to provide interfaces between distributed ledgers and real-world data, such as Town Crier[7] and Oraclize (www.oraclize.it). Otherwise, both Corda and Hyperledger Fabric attempt to be modular in their choice of protocols (see Item 6), which means that they are designed to interoperate across different usages of the overall system, but to the best of our knowledge, there has been little attempt among existing solutions to interface with one another.

4. Scalability: Why Store Every Transaction?

The problem. Scalability can mean a lot of things. Item 1 discusses the need to scale the time it takes to process individual transactions with the computational power that is added to the network. Another aspect of scalability is that, as the system becomes more popular, it should not significantly increase the storage load placed on users of the system, as this deters participation and increases the barrier to entry. For integrity purposes, however, it is important in systems in which we expect full public verifiability (discussed further in Item 1) to not delete entries from the ledger. This results in, for instance, the Bitcoin blockchain requiring currently over 130 GB to store, as it contains every transaction that has ever taken place, and growing at a rate of roughly 144 MB per day (http://blockchain.info/charts/blocks-size). The main problem thus lies in how to provide a balance between these two (seemingly contradictory) requirements.

In certain applications, it may be sufficiently important to provide full auditability (for example, to satisfy regulatory requirements) that we cannot help but increase the storage load of participants. Especially in consumer applications, however, a relatively compelling case can be made that it is not actually necessary to store every transaction, as—for example—we are unlikely today to want or need to meaningfully examine someone's coffee purchases from early 2010. In fact, retaining such information over a long period of time is clearly at odds with the privacy of users (Item 2).

In addition, in a slightly altered trust model, more "casual" users may be willing to offload the work of auditing the system to more "archival" users; thus, while archival users would be required to store the entirety of the ledger, casual users could store only the information needed to check the validity of their own personal transactions.

Potential and developing solutions. The main innovation that allows Bitcoin wallets to run on smartphones—crucially, without storing the full Bitcoin ledger—is the idea of an SPV (Simplified Payment Verification) client, which is analogous to the casual user mentioned above. These clients retrieve from archival peers and store only the headers of blocks along the blockchain, rather than their full contents, and rely on these headers to check for double spending. More specifically, with the help of an archival (or "full") node, an SPV client checks only that a given transaction is included in the blockchain; it thus reveals the transactions in which it is interested to archival nodes and trusts the miners to prevent double spending from being included. While such clients have gained wide adoption, their security—and in particular the privacy and trust issues mentioned—is imperfect and somewhat poorly understood.

Another major development that would help prevent excessive transactions from being included in the ledger is the Lightning network, which is an example of a more general (and increasingly active) line of research on payment channels (http://en.bitcoin.it/wiki/Payment_channels). To open such a channel, two parties can create a transaction that effectively serves to jointly lock up a certain amount of funds. The parties can then transact an arbitrary number of times between themselves (crucially, without placing any transactions on the ledger), and when they are ready to close the channel—either because they have depleted their funds or no longer expect to transact—they can place another transaction on the blockchain that serves this purpose. The storage load on the blockchain is thus significantly reduced, as a single pair of transactions represents an arbitrary number of transfers. Furthermore, because this process can be designed to handle disputes, it still doesn't require the two parties to trust each other. Again, while this approach seems promising and has attracted significant attention, the security and privacy aspects of using such channels are still not well understood.

Finally, another promising approach is the idea of *sharding* the ledger, as discussed in Item 1. If implemented in a certain way, participants do not need to store (or even hear about) transactions that are irrelevant to them; for example, someone doesn't need to store everyone's morning coffee purchases, just their own. While this significantly reduces the storage requirements of an individual user, it is still important to understand the practical costs of storing what is ultimately a monotonically growing ledger.

3. Cost-Effectiveness: What Is the Cheapest Way?

The problem. People like to complain a lot about Bitcoin and its proof-of-work-based consensus protocol. They compare the electricity it uses to the electricity used by whole nations, or at least by large power plants. They call it an "environmental disaster." While many of these arguments are quite overblown, the fact remains that proof of work is indeed very expensive,

and it would of course be a good thing if the same result could be achieved in a cheaper way.

To this end, there have been a huge number of alternative consensus protocols proposed, and even used in alternative cryptocurrencies. For example, proof of stake is a protocol in which, rather than preventing Sybil attacks on the mining process by requiring participants to consume some expensive resource, as is done in proof of work, so-called validators prove their "stake" in the system, in the form of an investment they have made that provides an economic disincentive for them to misbehave by forming bad blocks. Peercoin (http://peercoin.net), Nxt (http://nxt.org), and BlackCoin (http://blackcoin.co) use a version of proof of stake in which the investment is (roughly) coins that have been accumulated over time, and Ethereum has a planned transition (that is, hard fork) to its own version of proof of stake, Casper, within the next two years, in which the investment is a security deposit that locks up some of the validator's coins for a certain period of time. Intel's Sawtooth Lake platform uses proof of elapsed time (PoET), which relies on its SGX architecture. One of the most popular Bitcoin-based proposals, Bitcoin-NG,[8] proposes isolating the usage of proof of work to elect a periodic "leader" who can then quickly (that is, without performing large amounts of computation) bear witness to individual transactions.

Potential and developing solutions. In the distributed ledger research community, this seems to be the issue being explored most actively. In addition to the protocols mentioned above, there are dozens more articles with their own proposals, both for the protocols mentioned above (especially proof of stake, which seems to be a sort of "holy grail" as of this writing) and for new and increasingly exotic ones. For now, however, the vast majority of cryptocurrencies use some form of proof of work.

If one expands outward to general distributed ledgers, the list of alternative consensus protocols grows and begins to include more classical consensus protocols: two-phase commit, PBFT,[9] and so on. These tend to be significantly more efficient than the "proof-of-X" protocols used in cryptocurrencies, but with the tradeoff that they require a fixed set of known participants. As in Item 8, what is needed is a way to understand this tradeoff, provide meaningful comparisons within this growing landscape of consensus protocols, and understand the benefits that each provides in a given setting.

2. Privacy: How to Protect Data?

The problem. While significant research has focused on the anonymity of users in cryptocurrencies (that is, protecting the identity of participants), little research has focused on their privacy. For example, Bitcoin users are technically "pseudonymous," as their on-chain identities are not inherently linked to their real-world identity, but the details of each transaction—for example, the amount of bitcoins being sent—are still completely transparent. If we consider more expressive platforms and more exotic use cases, the desire to store health records on a distributed ledger means transactions would contain information about a patient's name (that is, their real-world identity), their age, the nature and justification of medical procedures, and so on. While a naive solution would be to encrypt this data and provide decryption keys only to the necessary parties, the fact is that encryption schemes, as with all cryptographic primitives, break or are compromised (see Item 6), so this does not provide a long-term solution for privacy.

Even with respect to anonymity, there's still a lot of work to do. A long line of research has demonstrated the limitations of Bitcoin's anonymity,[10] and while emerging platforms such as Monero[11] and Zcash[12] provide strong theoretical guarantees of improved anonymity, they have not yet been analyzed empirically. It is thus possible that they have their own set of weaknesses that fall outside the abstract protocol specification but could emerge with increased adoption and patterns of usage.

Potential and developing solutions. One of the simplest solutions is to send transactions to only those participants with whom you trust the associated details; this is akin to the sharding-based solutions we explore in Item 1. Even here, however, it is still important to consider privacy, as transactions may be shared beyond the initial set of participants, or it may be necessary to hide certain details from one participant but reveal them to another.

One general solution that has been proposed is Hawk,[13] in which users can hide the details of their transactions but still convince the rest of the network that the transactions are valid. While useful, Hawk is specific to Ethereum and makes use of fairly advanced cryptography. Thus, it is still very much an open question how to achieve privacy in a lightweight and flexible manner for general distributed ledgers.

1. Scalability: Do We Need Full Agreement?

The problem. Arguably the biggest hurdle for fully distributed ledgers is the insistence that every node in the network needs to agree on the full state of the entire ledger. Aside from the issues with this approach raised earlier (for example, in Item 4), this means that distributed ledgers cannot and do not scale in terms of their ability to process growing numbers of transactions (throughput) while still ensuring that users do not have to wait for their transactions to be included (latency). In other words, the more computational power that joins the network, the worse it will perform in terms of throughput and latency. It is precisely because of this requirement

that every node must agree on every transaction, as it means the more transactions are in the system, the longer the nodes must wait for them to flood the network.

Because this insistence on full replication thus violates one of the most basic properties of distributed systems, we might naturally ask ourselves: Why do it in the first place? One of the primary benefits of cryptocurrencies, enabled by this requirement, is the idea of full public verifiability: any participant can verify for themselves the correct functioning of the system by, for example, replaying all transactions and ensuring that any agreed-on rules haven't been violated. If only certain nodes agree on certain parts of the ledger, then there is no one participant that can satisfy this notion of verifiability. To allow for increased throughput and decreased latency, one must therefore provide a balance between avoiding full replication—which is often accomplished using sharding, in which each participant processes transactions only within a given shard—and enabling at least some degree of openness and verifiability.

Potential and developing solutions. This topic has received significant attention, and a number of academic proposals adopt some form of sharding.[14,15] Many industrial proposals similarly adopt a type of sharding; for example, in Corda, participants need to achieve consensus only on transactions that are directly relevant to them, and in Certificate Transparency there is no global consensus on the contents of the ledger. While these approaches achieve significantly better scalability, they raise questions about verifiability that fully decentralized solutions do not. For example, if only certain participants see certain transactions, how can other participants tell that their transactions obey the global set of rules? In the absence of such a global set of rules, what meaningful notions of integrity can we even satisfy?

In this last item, as with all the items on the list, we again see that each platform does not provide one uniquely perfect solution, but rather a set of tradeoffs that—to come full circle back to Item 10—must ultimately be balanced according to the individual use case in which the technology will be used. ■

Acknowledgments

This work was generously supported in part by R3 and by EPSRC Grant EP/N028104/1. Thanks are also due to the anonymous reviewers for their helpful feedback.

References

1. S. Abramova and R. Böhme, "Perceived Benefit and Risk as Multidimensional Determinants of Bitcoin Use: A Quantitative Exploratory Study," *Proceedings of the 37th International Conference on Information Systems* (ICIS 16), 2016.
2. V. Lehdonvirta, "The Blockchain Paradox: Why Distributed Ledger Technologies May Do Little to Transform the Economy," 2016; http://blogs.oii.ox.ac.uk/policy/the-blockchain-paradox-why-distributed-ledger-technologies-may-do-little-to-transform-the-economy.
3. A. Miller et al., "Nonoutsourceable Scratch-Off Puzzles to Discourage Bitcoin Mining Coalitions," *ACM CCS*, 2015, pp. 680–691.
4. L. Luu et al., "Smart Pool: Practical Decentralized Pooled Mining," *Proceedings of the 26th USENIX Security Symposium*, 2017.
5. A. Gervais et al., "On the Security and Performance of Proof of Work Blockchains," *ACM CCS*, 2016, pp. 3–16.
6. M. Chase and S. Meiklejohn, "Transparency Overlays and Applications," *ACM CCS*, 2016, pp. 168–179.
7. F. Zhang et al., "Town Crier: An Authenticated Data Feed for Smart Contracts," *ACM CCS*, 2016, pp. 270–282.
8. I. Eyal et al., "Bitcoin-NG: A Scalable Blockchain Protocol," *Proceedings of the 13th Symposium on Networked Systems Design and Implementation* (NSDI 16), 2016.
9. M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," *Proceedings of the Third Symposium on Operating Systems Design and Implementation* (OSDI 99), 1999, pp. 179–186.
10. J. Bonneau et al., "SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies," *IEEE Symposium on Security and Privacy*, 2015, pp. 104–121.
11. S. Noether, A. Mackenzie, and the Monero Research Lab, "Ring Confidential Transactions," *Ledger*, vol. 1, 2016, pp. 1–18.
12. E. Ben-Sasson et al., "Zerocash: Decentralized Anonymous Payments from Bitcoin," *IEEE Symposium on Security and Privacy*, 2014, pp. 459–474.
13. A.E. Kosba et al., "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts," *IEEE Symposium on Security and Privacy*, 2016, pp. 839–858.
14. G. Danezis and S. Meiklejohnm, "Centrally Banked Cryptocurrencies," *NDSS*, 2016.
15. L. Luu et al., "A Secure Sharding Protocol for Open Blockchains," *ACM CCS*, 2016, pp. 17–30.

Sarah Meiklejohn is a Reader in Cryptography and Security at University College London. She has broad research interests in computer security and cryptography and has worked on topics such as anonymity and criminal abuses in virtual currencies, anonymous credentials, and understanding the interface between cryptographic primitives and their mathematical underpinnings. Previously, Meiklejohn received a PhD from University of California, San Diego, in May 2014 under the guidance of Mihir Bellare and Stefan Savage, as well as an ScB in mathematics in 2008 and an ScM in computer science in 2009, both from Brown University. Contact at s.meiklejohn@ucl.ac.uk.

A First Look at Identity Management Schemes on the Blockchain

Paul Dunphy and Fabien A.P. Petitcolas | OneSpan Innovation Centre

We introduce the emerging landscape of distributed ledger technology (DLT)-based identity management (IdM) and evaluate three representative proposals—uPort, ShoCard, and Sovrin—using the analytic lens of a seminal framework that characterizes the nature of successful IdM schemes.

Twenty-four years have passed since Peter Steiner first showed the world that "on the Internet, nobody knows you're a dog," yet that famous drawing still stands to illustrate the challenge to identify individuals online. Today, we are very far from the public directory vision of the inventors of public-key cryptography in the 1970s or the grand scheme of hierarchical certification envisaged in the 1980s. Identity management (IdM) on the Internet still relies on what Cameron called a decade ago a "patchwork of identity one-offs,"[1] comprising several types of IdM systems that are restricted to specific domains and do not interact much with one another. Centralized models of IdM currently face challenges due to the increasing regularity of data breaches that lead to reputation damage; identity fraud; and above all, a loss of privacy for all concerned. These recurring events highlight a lack of control and ownership that end users experience with their digital identities.[2–4]

The investigation of alternative approaches to IdM is being led by initiatives that seek to expand the trustworthiness and reach of digital forms of identity. The United States' National Strategy for Trusted Identities in Cyberspace (NSTIC) aims to accelerate the development of novel technologies that can increase trust in online transactions.[5] In addition, ID2020 seeks to leverage emerging digital technologies to expand the reach of legal identities (mirroring the United Nations' goals to "provide [by 2030] legal identity for all, including birth registration"[6]). The emergence of Bitcoin[7] has also inspired fresh thinking about digital identity due to its underpinning distributed ledger technology (DLT) not requiring a central authority to validate transactions of its native cryptocurrency. Thus, a globally decentralized network is able to reach consensus on the current state of its book of transactions, the "ledger." The distributed ledger itself is an append-only shared record of transactions that is maintained by entities on a peer-to-peer network, whereas the often-cited "blockchain" is a cryptographic data structure that is often instrumental in DLTs and is constructed through cryptographic hashing of blocks of transactions.

Given that DLT is suited to ensuring consensus, transparency, and integrity of the transactions that it contains, a number of benefits of applying DLT to IdM have already been proposed:

- *Decentralized*—Identity information is referenced by a ledger that no single central authority owns or controls.

 Copublished by the IEEE Computer and Reliability Societies

- *Tamper resistant*—Historical activities in the DLT cannot be tampered with and transparency is given to all changes to that data.
- *Inclusive*—New ways to bootstrap user identity can be conceived that expand the reach of legal identities and reduce exclusion.
- *Cost effective*—Shared identity information can lead to cost savings for relying parties along with the potential to reduce the volume of personal information that is replicated in databases.
- *User control*—Users cannot lose control of their digital identifiers if they lose access to the services of a particular identity provider/broker.

However, given these proposed benefits of incorporating DLT into future IdM schemes, is the path to new forms of DLT-based IdM really "inevitable"?[2]

Identity Management on the Blockchain?

IdM encompasses the processes and policies involved in managing the life cycle of attributes in identities for a particular domain.[8] Most IdM schemes today are centralized where a single entity such as an organization owns and controls the system. However, the identities themselves can have a scope that goes beyond single organizations, as when governments issue national identity cards for use with multiple organizations. In federated identity systems, users can use identity information established in one security domain to access another. Single sign-on schemes, such as Facebook Connect, can work this way. User-centric identity management places administration and control of identity information directly into the hands of individuals. Examples include network anonymization tools (for example, Tor and I2P) that minimize disclosure of personal information and password managers (for example, 1Password and LessPass) that securely keep track of different website credentials.

Despite the different approaches, one function that is fundamental to IdM is securely binding together an *identifier*—a value that unambiguously distinguishes one user from another in a particular domain—and *attributes* (sometimes called certifications or claims)—entitlements or properties of a user such as name, age, or credit rating. The first steps taken to tailor the use of DLT for establishing secure and decentralized identifier–attribute mapping were in the design of Namecoin: the longest surviving software fork of Bitcoin. Namecoin provides a human-readable, decentralized, and secure namespace for the ".bit" web domain. This achievement contradicted conventional wisdom that a naming system exhibiting all three characteristics of human readable, decentralized, and secure namespace could not be designed.[9] Blockstack[4] has extended Namecoin's scheme to create a decentralized public-key infrastructure (PKI): it registers bindings between a public key and a human-readable identifier.

Recently, several decentralized identity schemes have emerged that extend beyond naming and aim to provide a more complete suite of IdM functions. However, until now, there has been no evaluation of these proposals. We were interested in whether DLT-based IdMs have the potential to go beyond previous approaches or would simply create new "identity one-offs."

Approach

We started our inquiry by searching for blueprints of DLT-based IdM proposals that were technically scrutable (for instance, white papers and open source software). We excluded schemes that provided only naming and found that all fell into one of two categories:

- *Self-sovereign identity* is owned and controlled by a user without the need to rely on any external administrative authority and without the possibility that this identity can be taken away. This can be enabled by an ecosystem that facilitates the acquisition and recording of attributes, and the propagation of trust among entities leveraging such identities. Examples include Sovrin, uPort, and OneName.
- *Decentralized trusted identity* is provided by a proprietary service that performs identity proofing of users based on existing trusted credentials (for instance, a passport) and records identity attestations on a DLT for later validation by third parties. Examples include ShoCard, BitID, ID.me, and IDchainZ.

In this article, we focus on three particular DLT-based IdM schemes: uPort, ShoCard, and Sovrin. We chose these three schemes because, individually, they serve as key exemplars of the prevalent design decisions and challenges found in their respective genres, and together serve a similar purpose for the broader landscape of DLT-based IdM. In addition, they have provided the most technical detail of their scheme designs and are either underpinned by sizable online communities or have notable venture capital funding.

There is no definitive criterion to evaluate IdM schemes, so to generate early insights about individual schemes, we leveraged an evaluation framework known as the "laws of identity,"[1] which serve to pinpoint the successes and failures of digital identity systems. It is a widely known framework and represents a full spectrum of IdM concerns, encompassing security, privacy, and user experience. Furthermore, the laws provide an inherent flexibility, which is ideal for application to the

heterogeneous and early-stage DLT-based IdM schemes we considered. The laws themselves are as follows:

1. *User control and consent*—Any information that identifies the user should be revealed only with that user's consent.
2. *Minimal disclosure for a constrained use*—Identity information should be collected only on a "need-to-know" basis and kept on a "need-to-retain" basis.
3. *Justifiable parties*—Identity information should be shared only with parties that have a legitimate right to access identity information in a transaction.
4. *Directed identity*—Support should be provided for sharing identity information publicly or in a more discreet way.
5. *Design for a pluralism of operators and technology*—A solution must enable the interworking of different identity schemes and credentials.
6. *Human integration*—The user experience must be consistent with user needs and expectations to enable users to understand the implications of their interactions with the system.
7. *Consistent experience across contexts*—Users must be able to count on a consistent experience across different security contexts and technology platforms.

In the text that follows, where we refer to a specific law, we use bracket notation to reference the law number (for instance, (1) or (5)).

If an attacker can compromise a uPort application and replace trustees unnoticed via the controller, the uPortID is compromised permanently.

uPort

uPort is an open source decentralized identity framework that aims to provide "decentralized identity for all."[3] Its use case is IdM for next-generation decentralized applications on the Ethereum DLT and for traditional centralized applications such as email and banking.

Design

A uPort identity is underpinned by the interactions between Ethereum *smart contracts*: bespoke code that can regulate the movement of data and ether (the native cryptocurrency) on Ethereum. Smart contracts are uniquely addressed by 160-bit hexadecimal identifiers and, when invoked, are executed by the Ethereum Virtual Machine (EVM) installed on every Ethereum node. Two smart contract templates designed by uPort's creators comprise each uPort identity: *controller* and *proxy*. To create a new identity, a user's uPort mobile application creates an asymmetric key pair and sends a transaction to Ethereum that initiates the creation of a new controller that stores a reference to the public key. Then, a new proxy is created that contains a reference to the just-created controller contract; only the controller can invoke functions of the proxy, a constraint that is specified in the controller and enforced by the EVM. The address of the proxy comprises the unique *uPort identifier* (uPortID). A user is free to create multiple uPortIDs. Figure 1a provides an overview of an interaction between a uPortID and the smart contract of a decentralized service on Ethereum.

The private key associated with a uPortID is stored only on the user's mobile device. Therefore, an important aspect of uPort usability is its key recovery protocol in the event of loss or theft of the user's mobile device. For key recovery, users must nominate trustees, who can trigger a vote to set a new public key via the controller; once a quorum is reached, the controller replaces the lost public key with a new nominated key by invoking a dedicated function of the proxy. This process enables the user to maintain a persistent uPortID even after the loss of cryptographic keys.

A final aspect of the uPort scheme is its support for securely mapping identity attributes to a particular uPortID. The uPort *registry* is a smart contract that stores the global mapping of uPortIDs to identity attributes. Any entity can query the registry; however, only the owner of a specific uPortID can modify its respective attributes. Due to the cost of storing large volumes of data in a smart contract, only the hash of the JSON attribute structure is proposed to be stored in the registry. The data itself is stored on IPFS: a distributed file system where a file can be retrieved by its cryptographic hash.

Analysis

uPort has no central server and does not authenticate the owner of a uPortID; this passes the risk of unauthorized access to the local authentication methods on the user's mobile device. While the social recovery protocol provides one method to recover ownership of a lost or compromised uPortID, the trustees themselves could be one vector of attack because their own uPortIDs are openly linked to the user's uPortID; this transparency provides opportunities for collusion against a specific uPort user. If an attacker can compromise a uPort application and replace trustees unnoticed via the controller, the uPortID is compromised permanently. So while uPort does place more control over uPortIDs in the hands of its users—a

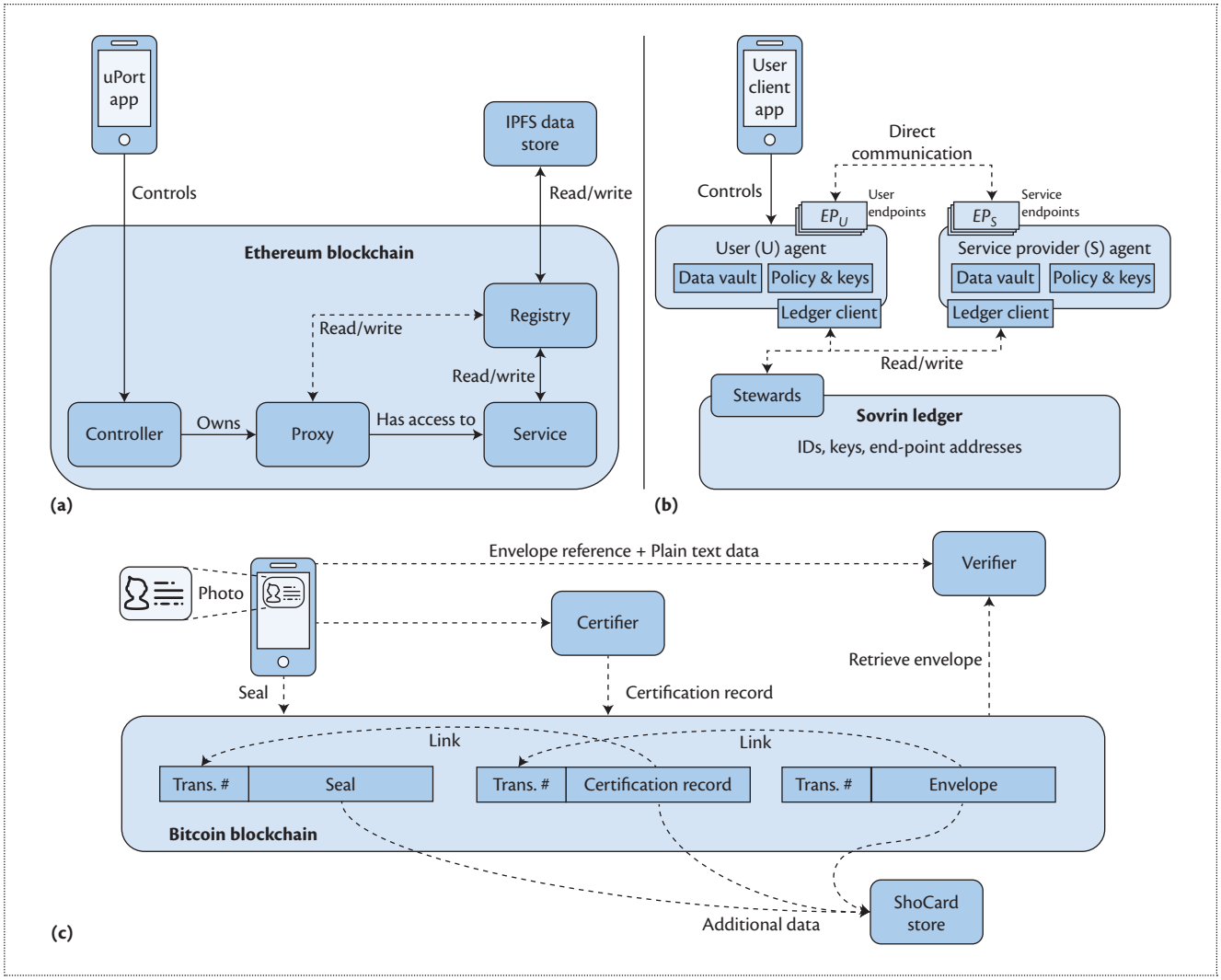


Figure 1. An overview of the key components of the (a) uPort, (b) Sovrin, and (c) ShoCard systems. In uPort, when interacting with a service hosted on the Ethereum network, the proxy can update the user data stored on a file system external to Ethereum, while the service used may read from it. At the base of Sovrin is a permissioned ledger. Only stewards that legally abide by the Sovrin Trust Framework can write to the ledger. Users can interact with the network through a client app. To be always accessible, users and organizations on Sovrin rely on agents that are addressable network points. Identifiers, keys, and endpoint addresses are stored on the ledger while attributes are stored off the ledger. ShoCard uses the Bitcoin blockchain to store and link together signed cryptographic hashes of identity data. The plaintext data is stored on a dedicated server in encrypted form.

plus for (1)—a layer of added complexity and responsibility is inevitably handed to users.

uPort does not require personal data disclosures to bootstrap a uPortID for a constrained use and also respects privacy in terms of the lack of inherent linkability between uPortIDs (2). However, the registry (if used) represents a point of centralization that can be probed for information. So while specific attributes within the attribute data structure can be individually encrypted, the overall JSON data structure is still visible, which could leak metadata about specific attributes or relationships with identity providers or relying parties. Thus there is a chance that over-reliance on the registry can compromise privacy (3).

A commerce application can widely advertise its uPortID, but uPort provides no public directory to look up uPortIDs from arbitrary search criteria. Discreet disclosure of a uPortID is possible if a user creates new uPortIDs for each new relying party that they encounter (4). However, because a uPortID equates to a smart contract, an honest but curious Ethereum node could discover even nondisclosed uPortIDs through analysis of the

smart contract code stored at a given address to determine if it is a uPort template. More work is needed to discover whether nondisclosed uPortIDs are private in practice.

uPort does not perform any identity proofing but instead provides a framework for users to gather attributes from an ecosystem of identity providers; uPort simply specifies the format of attributes that are stored in its registry. But as a consequence of the uPortID owner alone having write access to their own respective part of the registry, a user can selectively discard negative attributes that they are given, for example, a low credit score, a criminal conviction, and so on (5).

The mobile application of uPort provides a consistent user experience across all usage contexts (7) due to the scanning of a QR code being relied on to initiate interactions with a relying party. However, the in-app education does not convey privacy implications of storing representations of personally identifiable information on a blockchain (6). The area of user education will become pressing in this context as legislation such as the European General Data Protection Regulation (GDPR) come into force.

Sovrin

Sovrin is an open source identity network built on permissioned DLT that stores identity records.[2] Sovrin is public, but only trusted institutions, called *stewards*—which could be banks, universities, governments, and so on—can run nodes that take part in consensus protocols; thus, the ledger is *permissioned*. The nonprofit Sovrin Foundation ensures the proper governance of the stewards and their respect of a legal agreement called the Sovrin Trust Framework. Sovrin provides the code base to the Hyperledger Indy project.

> **Users must rely on agencies that will act on their behalf in the Sovrin network and on the stewards maintaining the distributed ledger. Depending on the choice of agent and its implementation, a lot of information could potentially be in the hands of the agency.**

Design

Sovrin enables a user to generate as many identifiers as needed to keep contextual separation of identities for privacy purposes; each identifier is unlinkable and controlled by a different asymmetric key pair. Sovrin identifiers are managed by the user or an appointed guardian service and follow the Decentralized Identifier (DID) specification currently seeking Internet Engineering Task Force (IETF) standardization. A DID is a data structure containing the user identifier, cryptographic public key, and other metadata necessary to transact with that identifier.

The Sovrin architecture can be summarized by the components as shown in Figure 1b. The key element is the Sovrin ledger. This contains identity transactions associated to a particular identifier and is written, distributed, and replicated among the *steward nodes*, which run an enhanced version of the redundant Byzantine fault-tolerant protocol of Aublin and colleagues,[10] called Plenum, for consensus.

There are two important consequences to the choice of permissioned ledger in Sovrin's design. First, no expensive proof-of-work computation is required to reach consensus on the state of the ledger, significantly reducing the energy cost of running a node and dramatically improving transaction throughput. Second, trust on Sovrin relies on both people and code. Trust starts from the common root of trust formed by the globally distributed ledger, but as new organizations and users join the network, they can become *trust anchors* (that is, allowed to add more users and organizations); a "web of trust" is expected to evolve to support this decentralized network growth.

Users interact with Sovrin through a mobile application and control software *agents* acting on their behalf to facilitate interactions with other agents on the network. Agents are network endpoints that are always addressable and accessible. Users could run their agents on their own servers, but more likely, they will ask specialized intermediaries, *agencies*, to do that for them, like email systems. Agents also provide backup service and encrypted storage of attribute credentials. The format of these attribute credentials align with the emerging W3C Verifiable Claims standard for credentials verified by third parties.

The mobile application also helps users manage cryptographic keys, which are stored on the users' mobile device. As in uPort, Sovrin offers a mechanism for key recovery that relies on the user selecting a set of trustees. When requested to do so by the user, a specified quorum of trustees must sign a new identity record transaction that stewards must verify.

Analysis

Sovrin aims to equip users to fully control all aspects of their identity. Each user can selectively disclose attribute credentials that they hold to meet the identity validation requirements of a relying party (1). Also, the privacy that can be achieved in this process can be enhanced through the use of anonymous credential technology.

Although users can choose to store those attributes on the ledger, in general, they will prefer to use the storage capabilities of their mobile phone or their agent to transmit attributes to other parties through secure communication channels and use the ledger to identify the correct network endpoint to use. The use of attribute-based credentials allows users to reveal only information that is necessary (2). Verifying the party with whom data is shared remains a challenge, which is partly addressed through the web of trust, the governance of the Sovrin Foundation, and the reputation of the stewards.

Although there are no trusted third parties in the PKI sense on Sovrin, users must rely on agencies that will act on their behalf in the Sovrin network and on the stewards maintaining the distributed ledger. Depending on the choice of agent and its implementation, a lot of information could potentially be in the hands of the agency. However, as agencies are acting on behalf of the user, they have a "necessary and justifiable place" in the identity relationship (3).

Sovrin supports both omnidirectional and unidirectional identifiers (4): public organizations can decide to publish their full identity on the network, while users may choose to publish only identifiers and to use different identifiers and cryptographic key pairs with each party they interact with, avoiding emitting "correlation handles."

> **ShoCard's intermediary role does create uncertainty about the longitudinal existence of a ShoCardID; if the company ceased to exist, users of ShoCard would be unable to use the system with the certifications they had acquired.**

Today, Sovrin depends on a very small number of operators sharing the same implementation. As the systems gets traction, new agencies, and new stewards, will join. The Sovrin Foundation expects in particular to build a market of agencies that will compete on the features they offer, for instance, interfaces with other (existing) identity systems (5).

An important issue not yet addressed by the Sovrin developers is the user experience. The history of security offers several examples of smart cryptographic systems, which have never been deployed widely because users found it too cumbersome or difficult to understand—email encryption using PGP is a seminal example. So, human integration remains a big open question for Sovrin. Considering that Sovrin is still in the early development phase, evaluating it against laws (6) and (7) is tricky, but it is illustrative that much work has considered the scheme architecture design, but hardly any has considered the user experience.

ShoCard

ShoCard is a digital identity card on a mobile device that binds a user identifier, an existing trusted credential (for instance, passport or driver's license), and additional identity attributes together via cryptographic hashes stored in Bitcoin transactions.[11] ShoCard's primary use cases are verification of identity in face-to-face and online interactions.

Design

ShoCard uses Bitcoin as a timestamping service for signed cryptographic hashes of the user's identity information, which are mined into the Bitcoin blockchain. ShoCard incorporates a fixed central server as an essential part of its scheme; this server intermediates the exchange of encrypted identity information between a user and a relying party. The scheme relies on three phases: *bootstrapping*, *certification*, and *validation*. Figure 1c schematizes those phases.

Bootstrapping occurs at the creation of a new ShoCard. The ShoCard mobile application generates an asymmetric key pair for the user and scans their identity credentials using the device's camera. The scan and the corresponding data are encrypted and stored on the mobile device; the signed hash of this data is also embedded into a Bitcoin transaction for later data validation purposes. The resulting Bitcoin transaction number constitutes the user's ShoCardID and is retained in the mobile application as a pointer to the ShoCard *seal*.

Once a ShoCard is bootstrapped, the user can interact with service providers to gather additional attributes that rely on the seal in a process called certification. To associate certificates to a ShoCardID, an identity provider must first verify that the user knows both the data hashed to create the seal and the cryptographic key used to create its signature. In a face-to-face context, this can be achieved by the user providing the original identity data forming the seal from their mobile device, a digitally signed challenge, and the original trusted credential. The certificate takes the form of a signed hash of new attributes (and its associated ShoCardID) in a Bitcoin transaction created by the provider. The provider must share the Bitcoin transaction number, along with a signed plaintext of the new attributes, directly with the user. Because the user will later need to provide the attributes to relying parties and may not want to lose them if the mobile device

is lost, a ShoCard server offers storage for symmetrically encrypted certifications (known as envelopes). ShoCard never learns the encryption key, which enables the user to share certifications only with selected parties.

The validation phase occurs when a relying party must verify a certification to determine whether a user is entitled to access a service. To validate the envelope, the user must first provide the relying party with the envelope reference and its encryption key. After retrieving the envelope from the ShoCard servers, the relying party checks that:

- the envelope signature was produced with the same private key that signed the seal;
- the certification signature was created by a trusted entity and the plaintext certification corresponds to the one hashed and signed in the blockchain; and
- the textual details presented by the user in the pending transaction match those embedded in the seal.

Analysis

The ShoCard central server functions as an intermediary to manage the distribution of encrypted certifications between ShoCard users and relying parties. In this way, ShoCard bears less risk than if it stored and distributed plaintext identity data. Secure storage of identity information and appropriate sharing with relying parties is controlled by the end user (1). However, ShoCard's intermediary role does create uncertainty about the longitudinal existence of a ShoCardID; if the company ceased to exist, users of ShoCard would be unable to use the system with the certifications they had acquired. This makes ShoCard more centralized in practice than its open reliance on DLT might suggest.

Each ShoCard identity must be bootstrapped with an existing trusted credential, such as a passport or driver's license. Such an approach requires users to provide personal information from the outset in order to create a ShoCard seal. This may make ShoCard less attractive for low-value online accounts (2).

Because the user is in control of initiating sharing activities, and because ShoCard stores only encrypted data, there can be some confidence that only justifiable parties are involved in the identity data–sharing transaction. However, the ShoCard server may be able to associate a particular ShoCardID with requests made by relying parties, since envelopes must be retrieved from the ShoCard server by the relying party (3).

ShoCard supports only unidirectional identifiers and does not support a public registry of ShoCardIDs. Omnidirectional identifiers may be needed in the future to realize its vision of an ecosystem of reusable certifications (4).

ShoCard does support a multitude of different identity providers through its certification functionality, but those providers must create bespoke integration with ShoCard's own web services in addition to Bitcoin, which could be a barrier to uptake. The decision to leverage ShoCard in future applications can only be driven by positive perceptions of the trustworthiness of ShoCard's identity proofing of its users, and the resulting value of a ShoCard (5).

The scanning of identity documents and QR codes is a dominant interaction paradigm in the ShoCard user experience: it is simple and consistent (7). However, it is unclear what the user motivations would be to adopt this new type of digital identity and how users would be educated about the implications of referencing identity data on a blockchain (6). Users are also not supported with cryptographic key management.

> **While DLT applications often target the removal of the "middle man," this may not be a realistic goal in IdM applications due to the context of identity maintaining a profound need for trust.**

One final point concerns the overall deployability of ShoCard. Bitcoin transactions take on average 10 minutes to be mined into the blockchain, and waiting for six additional blocks to be mined is recommended before assuming the settlement of a transaction. This could bring the waiting time for settlement to one hour on average. In a context that requires real-time settlement of certifications, this speed could create challenges for the user experience and those who wish to build applications that leverage ShoCard.

Discussion

Table 1 summarizes each scheme that we evaluated with respect to each law of identity. An unshaded table cell indicates that we found evidence that a scheme complied with a specific law, and a shaded cell indicates that we currently see no evidence that a scheme complies with a specific law. We include a summary of Facebook Connect to provide contrast.

Decentralization That Relies on Centralization and Intermediaries

DLT is often seen as a remedy for system architectures dominated by central authorities and intermediaries. But while each DLT-based IdM scheme we looked at

Table 1. A summary of uPort, ShoCard, Sovrin, and their relation to Cameron's laws of identity.*

Law	uPort	ShoCard	Sovrin	Facebook Connect
1—User control and consent	User controls creation and disclosure of uPortIDs and can prove ownership of uPortID without a central authority. But attributes stored in registry may leak information.	User controls creation and disclosure of ShoCardIDs. Attributes are accessible to a relying party only by invitation of ShoCardID owner. But ShoCard servers are necessary part of attribute validation protocol.	By design, users can choose which DIDs are used and which attributes are revealed. A web of trust that could be reinforced by a reputation system helps protect users against deception.	Today, when using Facebook to log on to a service, the user can choose which data will be shared by Facebook with the relying party.
2—Minimal disclosure for a constrained use	Users do not need to disclose personal data in order to create uPort identifiers for low-value accounts.	ShoCardIDs are bootstrapped with a trusted identity document (for example, a government ID).	Support of anonymous credentials based on zero-knowledge proofs allows users to share the information "least likely to identity [them] across multiple contexts."[1]	A user can create an empty Facebook profile and progressively add identity information as needed.
3—Justifiable parties	The JSON structure of attributes in the registry is visible to all, which may leak information to an honest-but-curious attacker—even if encrypted.	ShoCardID is revealed to a relying party only at the invitation of the ShoCardID owner. ShoCard servers may learn identity of relying parties.	Attributes are accessible only to relying parties that the user chooses, and to the agencies entrusted to act on their behalf.	Facebook always has access to the data stored on a user's Facebook profile whether the data is public or private. Facebook also creates and processes its own attributes, for instance, relationships with friends.
4—Directed identity	Supports unidirectional sharing of identifiers between parties, but does not prevent entities broadcasting identifiers out of band, for instance, on websites.	Supports unidirectional sharing of identifiers between parties, but does not prevent entities broadcasting identifiers out of band.	Omnidirectional identifiers are supported.	Omnidirectional identifiers are supported. A user's Facebook profile can be made public or private, and profiles can be searched.
5—Design for a pluralism of operators and technology	Agnostic to the types of attributes that third party identity providers create, yet use of a specific data format is encouraged in the registry.	Supports parsing of existing trusted credentials, but relying parties must create bespoke integrations with ShoCard centralized servers for attribute validation.	Expects to build a market for intermediaries (agencies) between users and the Sovrin network. Some could be interfaces with other identity systems.	Only one identity provider: Facebook. Uses a bespoke method for authorization to applications. But Facebook has nearly 2 billion users.
6—Human integration	Provides a mobile application. Social cryptographic key recovery function shows promise. Unclear usability and user understanding of uPort privacy implications.	Provides a mobile application. The digital ID card metaphor is easy to understand. Unclear usability and user understanding of ShoCard privacy implications.	Implementation has been targeted so far toward the underlying technology, not the user experience. Unclear usability and user understanding of privacy.	Facebook is well known to users and usable interface to single sign-on is provided. However, users may be unaware of privacy implications of Facebook Connect.
7—Consistent experience across contexts	User interaction driven by the mobile application. Consistently follows a QR code–scanning paradigm for all uses.	User interaction driven by the mobile application. Consistently follows a QR code–scanning paradigm for all uses.	Not clear. This will highly depend on the market of implementations of mobile device clients for the Sovrin network.	Consistent experience via the "login with Facebook" button.

** Facebook Connect is provided for comparison. An unshaded table cell indicates that we found evidence that a scheme complied with a specific law, and a shaded cell indicates that we currently see no evidence that a scheme complies with a specific law.*

leverages techniques of decentralization to different degrees, this served mainly to reshape the role of centralization and intermediaries rather than eradicate them. For example, uPort's registry stores a secure mapping between uPortID and its attributes and also relies on central authorities as trusted attribute providers. The ShoCard central server is an intermediary that stores encrypted identity attributes and mediates between end users and relying parties. Sovrin on the other hand embraces an open ecosystem of intermediaries (for example, agencies and trust anchors).

So, while DLT applications often target the removal of the "middle man," this may not be a realistic goal in IdM applications due to the context of identity maintaining a profound need for trust. Of course, this need for centralization and intermediaries is not necessarily a bad thing: there are numerous examples of centralization and intermediaries serving essential functions in an industry (see, for example, SWIFT). Elements of needed centralization or intermediation in a decentralized IdM may comprise:

- capturing additional authentication factors from end users;
- backing up and recovering cryptographic keys;
- providing a secure namespace to facilitate lookup of entities and services;
- securely storing the information hash pre-images needed to validate digital signatures; and
- recovering compromised DLT-based identities.

The case of "The DAO" stands as an example of the risks of pursuing too much decentralization in a system design. The DAO was designed as an Ethereum smart contract–based autonomous venture capital company, but a flaw in its underlying code enabled an attacker to steal $50 million of the funding that it collected.[12] The research challenge for DLT applications in IdM is therefore to explore the balance between centralization and decentralization to create interoperable and privacy-respecting IdM that mitigates the risk of placing too much trust in any single authority.

Ecosystems of Shareable Identity Attributes—But Ad Hoc Trust

Support for the creation and sharing of identity attributes certified by third parties is a design feature of each scheme we evaluated. In ShoCard, third parties can certify attributes of an identifier; uPort and Sovrin support both self-attestation of attributes and those assigned by other entities.

Designing for reusable identity attributes aims to improve the granularity at which users can disclose identity information and promotes reuse of attributes. However, due to the lack of a central authority, trust of these attributes currently relies on ad hoc trust establishment and integration between organizations. ShoCard and Sovrin propose a "web of trust" as the means by which attributes can be trusted. However, the challenges to design a web of trust are widely known where the network size is unbounded: difficulty to quantify trust beyond a first-degree relationship especially if any entity can vouch for any other, poor density of trust anchors on the network, lost or expired private keys, slow propagation of endorsement between users, and so on. DLT does not address any of those challenges, but future research could focus on methods to achieve the building of trust and reputation between entities in the context of DLT identity attributes. This could be one way that DLT-based IdM responds to NSTIC[5] and delivers new interoperability in IdM.

> **There appears to be a widespread assumption that users are equipped to conduct effective cryptographic key management and would intuitively understand the implications of referencing identity attributes in a DLT.**

If It Isn't Usable, It Isn't Secure

Dhamija explains in her "Seven Flaws of Identity Management" article that for users "identity management is not a primary goal."[13] This has been reflected in the shrug that users have largely given to single sign-on solutions—the user-facing proposition of IdM. This suggests that future IdM schemes with a novel technological underpinning but developed with the same blueprint of end user interaction are unlikely to create widespread uptake. A principal tenet of human–computer interaction is to design systems that respond to empirical evidence of challenges faced by end users. So far, we have seen that none of the schemes we evaluated are accompanied by a novel evidence-based vision of user interaction; furthermore, the perennial challenge to provide usable end user key management[14] is largely unaddressed. Recent research has suggested that key management remains a principal source of concern for users of Bitcoin.[15] While the promising concept of key recovery was proposed in uPort and Sovrin, approaches to digital identity that remove central authorities and depend on effective key management strategies from their users create the risk that nontechnical users will be

alienated by the technology, and when things go wrong, those users will be unable to recover resources or reputation attached to lost keys.

Distributed ledger technology is not a silver bullet solution for identity management. Our application of Cameron's evaluative framework provides an early glimpse of the current strengths and limitations of applying DLT to IdM. Future work in this nascent research area faces two particular hurdles.

First, there is a noticeable lack of contextual understanding relating to the user experience elements of the schemes we encountered. Usability is a particularly pressing unknown because there appears to be a widespread assumption that users are equipped to conduct effective cryptographic key management and would intuitively understand the implications of referencing identity attributes in a DLT.

Second, there is a tightening regulatory landscape for storing and processing personal data. For example, the GDPR grants end users new powers over personal data and places new obligations on data controllers and processors. This creates a challenge for the design of identity-focused immutable ledgers that reference personal data and that provide inherent transparency to data that they store.

Delaying the advance of new approaches to secure and trusted identities on the Internet is said to be an unacceptable course of action by the United States' NSTIC strategy.[5] This might be due to the concern that the online adage that "on the blockchain, nobody knows you're a fridge" may soon replace the prescience of Steiner's original cartoon. ■

References

1. K. Cameron, "The Laws of Identity," Microsoft Corporation, 5 Nov. 2005.
2. A. Tobin and D. Reed, "The Inevitable Rise of Self-Sovereign Identity," The Sovrin Foundation, 29 Sept. 2016.
3. C. Lundkvist et al., "uPort: A Platform for Self-Sovereign Identity," 21 Feb. 2017.
4. M. Ali et al., "Blockstack: A Global Naming and Storage System Secured by Blockchains," *2016 USENIX Annual Technical Conference* (USENIX ATC 16), 2016, pp. 181–194.
5. "National Strategy for Trusted Identities in Cyberspace: Enhancing Online Choice, Efficiency, Security, and Privacy," The White House, 2011.
6. "Transforming Our World: The 2030 Agenda for Sustainable Development," United Nations, 2015.
7. S. Nakamoto, "Bitcoin: A Peer-To-Peer Electronic Cash System," 2008.
8. "ISO/IEC 24760-1—Information Technology—Security Techniques—A Framework for Identity Management—Part 1: Terminology and Concepts," ISO/IEC, 2011.
9. Z. Wilcox-O'Hearn, "Names: Distributed, Secure, Human-Readable: Choose Two," 20 Oct. 2001.
10. P.L. Aublin, S.B. Mokhtar, and V. Quéma, "RBFT: Redundant Byzantine Fault Tolerance," *IEEE 33rd International Conference on Distributed Computing Systems* (ICDCS), 2013, pp. 297–306.
11. Travel Identity of the Future—White Paper, SITA, ShoCard, May 2016.
12. "Not-So-Clever Contracts," *The Economist*, 30 June 2016.
13. R. Dhamija and L. Dusseault, "The Seven Flaws of Identity Management: Usability and Security Challenges," *IEEE Security & Privacy*, vol. 6, no. 2, 2008, pp. 24–29.
14. A. Whitten and J.D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," *8th USENIX Security Symposium*, 1999, pp. 169–183.
15. S. Eskandari et al., "A First Look at the Usability of Bitcoin Key Management," *NDSS Workshop on Usable Security* (USEC 15), 2015.

Paul Dunphy is the lead researcher on the distributed ledger technology research theme at OneSpan Innovation Centre in Cambridge, UK. Prior to joining OneSpan, he spent time at Atom Bank: the UK's first bank to deliver services entirely via mobile applications that pioneered the use of mobile biometrics. He joined during Atom's start-up phase and shaped the successful first launch of its mobile applications, which in total have processed close to £1 billion in customer deposits. He completed a Microsoft Research–funded PhD at Newcastle University (UK) where his thesis focused on usable, secure, and deployable user authentication. He has also spent time leading research projects at Microsoft Research and Nokia Research. His research interests are broadly at the intersection of privacy and security with human–computer interaction. Contact at paul.dunphy@onespan.com.

Fabien A.P. Petitcolas is research manager at OneSpan Innovation Centre. Prior to joining OneSpan, Fabien spent 15 years at Microsoft where he took various roles. He first became a member of the Security Group at Microsoft Research where he focused on digital watermarking. He later became head of Microsoft Research's intellectual capital development programs, before becoming director for innovation at Microsoft Europe, supporting the company's presence in EU policy and political dialogue for and around innovation and R&D. Fabien received a PhD in computer science from the University of Cambridge under the guidance of Professor Ross Anderson FRS FREng. His research interests include information hiding, an area where he has authored several publications and books, and, more recently, security issues related to identity management and user authentication. Contact at fabien.petitcolas@onespan.com.

Tyranny of the Majority: On the (Im)possibility of Correctness of Smart Contracts

Lin Chen, Lei Xu, Zhimin Gao, Yang Lu, and Weidong Shi | University of Houston

Many consensus protocols are based on the assumption that participants are either "good" or "bad" but ignore the fact that they may be affected by direct or indirect economic interests involved in the corresponding smart contracts. We analyze consensus in decentralized environments and demonstrate that the system cannot guarantee correct execution results.

Since the development of Bitcoin, blockchain—the underpinning technology—has attracted growing attention from both industry and academia, and a wide range of applications has been developed on top of it. Blockchain provides a way for decentralized bookkeeping (a decentralized ledger) and achieves global consensus by preventing inconsistent records from being added to the system. By linking blocks with hash values and having each participant keep a local copy of the entire history, it is very difficult for an attacker to alter, modify, or remove existing data stored on the blockchain and decentralized ledger.

Besides storing data and transaction history, blockchain also provides the ability to execute any program and take custody of assets on the shared ledger in accordance with the results of the executed program using a decentralized model. Roughly speaking, each participant executes a program (chaincode or smart contract) locally and then submits the result to the blockchain. If a majority of the participants is honest, the correct execution result will be accepted into the blockchain as consensus and crystallized permanently as history of the immutable ledger. Along this direction, a blockchain-based smart contract system has been introduced and is believed to have the potential to revolutionize many areas such as finance, logistics, trade, supply chain, energy, and the Internet of Things, to name just a few.

Parties involved in a contract do not need to trust any single party. Instead, they only need to trust the system as a whole. Figure 1 demonstrates the basic concept of executing smart contracts on a blockchain-based system. In addition to the public ledger, each participating node has a smart contract–processing engine, which can execute contracts written using a predefined language. Briefly, a smart contract is executed in the following steps:

- *Contract submission.* Users involved in the contract achieve an agreement and prepare the contract—that is, a piece of code reflecting the business logic—and submit it to the system. Participants of the system

 Copublished by the IEEE Computer and Reliability Societies

then work to embed the smart contract into a block and add it to the blockchain.

- *Contract execution.* After a smart contract is added to the blockchain, each participant of the system runs it as a program and computes the result. In this process, all inputs that the smart contract depends on come from the blockchain or decentralized ledger.
- *Results submission.* Although each participant can submit his or her own result of a smart contract, all participants of the system work together to build a block to hold the result of the smart contract and add the block to the blockchain or ledger.

Outside of the basic procedure, different smart contract implementations may have their own special designs; for example, Ethereum requires the creator of a contract to attach some resources for the execution to discourage meaningless smart contracts. However, these system-specific features do not affect the abstract framework of blockchain-based smart contracts. It is often required that smart contracts be deterministic, and the results depend only on the contracts themselves and data stored on the ledger or blockchain (results of external or physical events are also recorded on the chain).

Although a smart contract can be an arbitrary piece of program that is executed in a decentralized manner, in this work, we consider an abstract smart contract system with the following characteristics:

- contracts involve financial and economic interest (custody of digital assets or cryptocurrency);
- the system is event driven;
- transaction history and results of contract execution are recorded using shared ledger;
- there is no centralized trusted party;
- there is no censorship or centralized policing mechanism; and
- the system is consensus based.

Consensus and Smart Contracts

The consensus mechanism plays an important role in the smart contract system at two times during the lifetime of a contract. First, when a smart contract is submitted to the system, all participants have to achieve a consensus on accepting the new block that contains the smart contract. Second, when multiple participants submit their results of a smart contract to the system, all participants need to achieve consensus on which result should be accepted.

The most popular consensus mechanism for blockchain is a *proof of work with longest chain* criterion. To produce a valid block, a participant needs to spend a significant amount of computation effort to solve a hard problem and attach a proof of such efforts to the block, for example, finding a pre-image of a hash function and providing the result to other participants. If two different blocks are produced and linked to the same predecessor, it will not be resolved immediately. Instead, each participant can keep both branches until one has more blocks than the other, and discard the shorter one.[1,2]

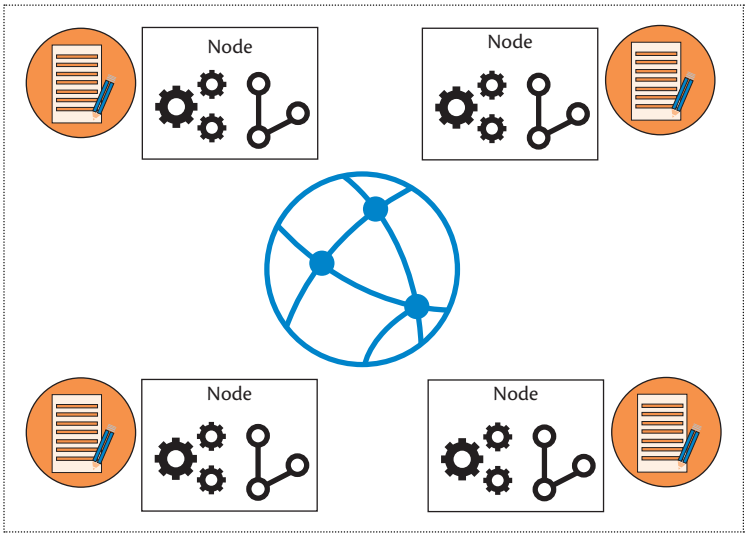


Figure 1. Smart contract system based on blockchain technology. Each participant executes a contract locally and submits the result to the system, and all participants act as brokers and work together to select a result (if there is more than one) to put into the decentralized ledger implemented using blockchain.

An alternative approach to achieve consensus in a decentralized environment is to leverage the classic Byzantine fault-tolerant (BFT) protocol, which was introduced by Lamport and colleagues.[3] In a basic scenario of the Byzantine general problem, there are both loyal generals and traitors in the system; these entities can exchange messages with each other to make the final decision. However, BFT requires a centralized identity management system to prevent Sybil attacks and does not work for the public blockchain environment directly.[4]

Limitations of Existing Model for Smart Contracts

Smart contracts in nature involve financial interests, which makes the process of achieving consensus more complex than pure cryptocurrency systems. In smart contract systems, nodes are potentially both brokers (judges to contract execution) and contract participants, which creates all kinds of possible conflict-of-interest scenarios regarding the judgment of contract execution outcome. The situation is even worse because participants are anonymous, which makes it impossible to detect and prevent conflicts of interest. Before

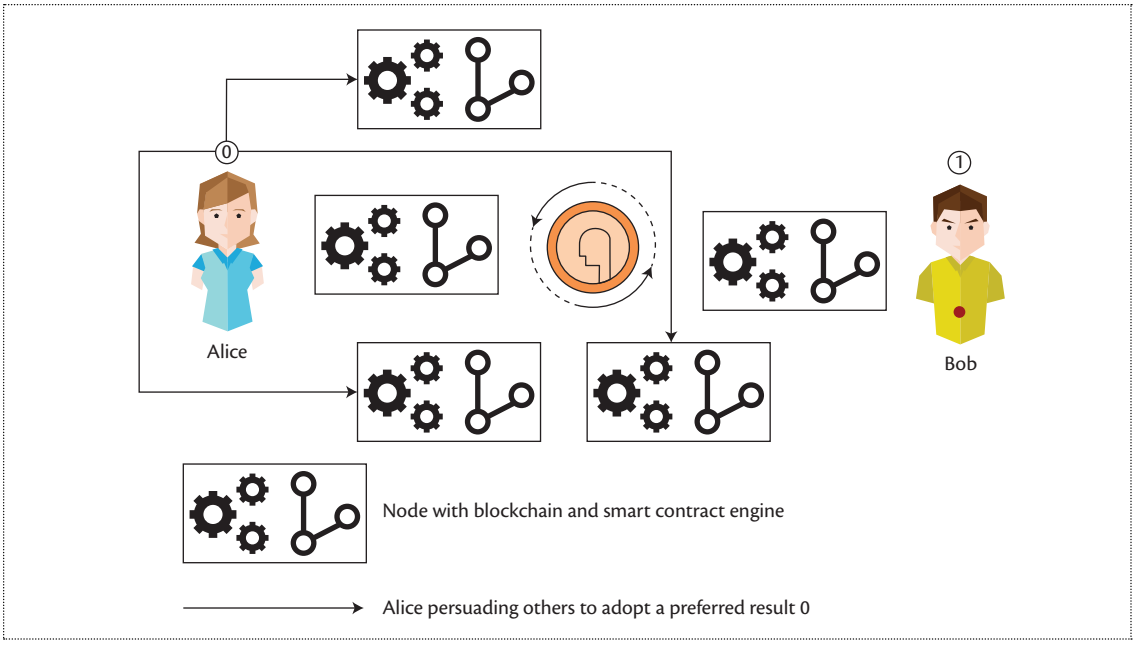


Figure 2. A smart contract involving flipping of a coin. The coin-flipping result determines the outcome of the smart contract; a participant can try his or her best to lead the system to accept his or her preferred result and maximize his or her profit.

discussing limitations of the existing consensus models, we first consider the following examples.

Example 1. Suppose we have two users, Alice and Bob, who want to have a contract based on the result of flipping a coin (0 or 1). The contract is very simple:

```
 if coin flipping result is 1
Alice loses x coins, and Bob wins x coins
 else
Bob loses y coins, and Alice wins y coins
 end if
```

More concretely, the coin-flipping result can be an event outcome in the prediction market that determines the profit of Alice and Bob. We remark that Alice and Bob can represent two groups of users other than two single users. When the smart contract is executed in a blockchain-based system, there should be only one correct result. (Assume that there is a decentralized algorithm to flip a coin over blockchain where the algorithm can be executed and the result can be verified by any participant of the system, and the outcome is unpredictable beforehand and deterministic when the algorithm is executed—for instance, computing a binary result based on the content of future x blocks.) Because Alice and Bob have their own preferred result, they can decide to accept their preferred result instead of true execution outcome. For instance, even if flipping coin returns 1, Alice can try her best to lead the system to adopt the block containing the wrong result 0 and maximize her interest, as shown in Figure 2. On the other hand, if the result of flipping a coin is 0, Alice would be happy to accept the correct result.

The above example involves only two participants. It is not difficult to imagine that in actual scenarios, a contract may involve a large number of participants (in certain cases, a significant percentage or even a majority of participants), where each participant has his or her own financially preferred outcome.

Example 2. If the number of participants who change their roles as impartial judges of a contract is relatively small, results of Byzantine general problem may still apply. However, this may not be always the case. A smart contract system likely contains a set of interdependent contracts, which can bind participants together and allow them to vote strategically on the contract execution outcome.

For instance, besides the example contract between Alice and Bob, each of them may have multiple contracts with other participants of the system. The execution result of a smart contract between Alice and Bob may well affect other participants whose financial interests are indirectly involved, as shown in Figure 3. There may exist a contract between Alice and Carol saying that Alice will pay back Carol if she earns profit from her contract with Bob. In this case, Carol has incentive to accept the outcome favoring Alice for the contract between Alice and Bob.

In complex scenarios with dependent contracts, the number of indirectly involved participants of a smart

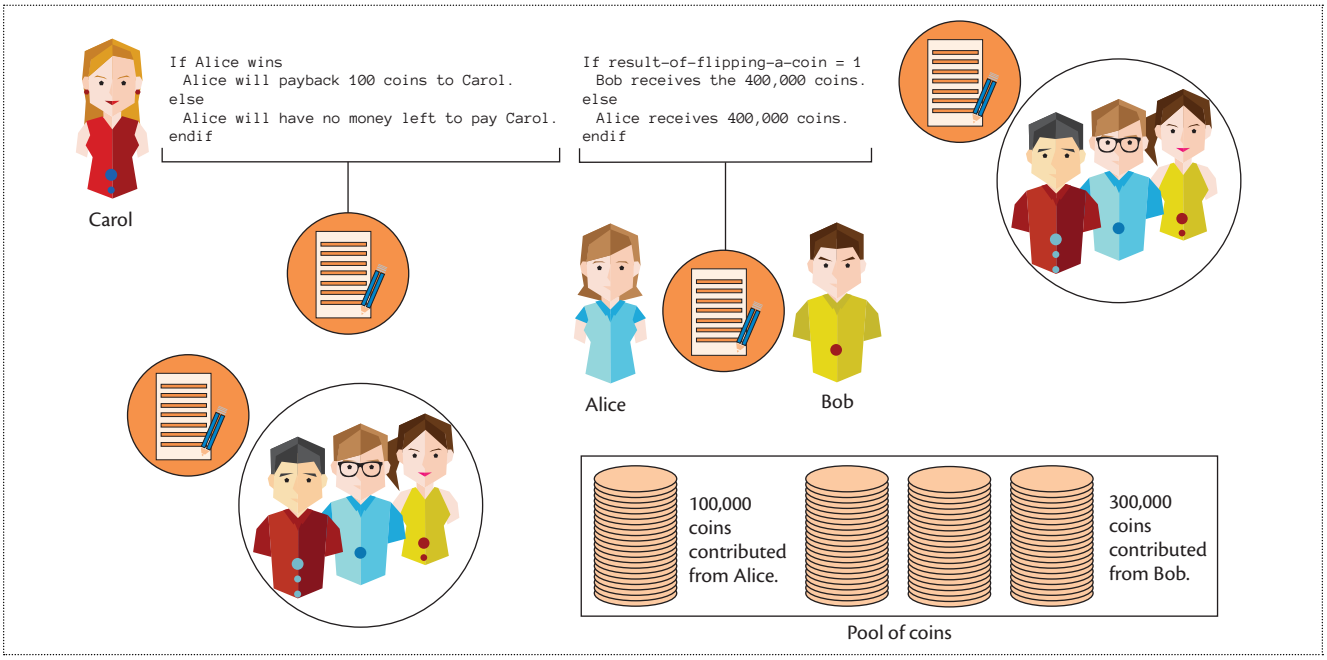


Figure 3. A smart contract involving a large number of indirect participants with economic interests tied to the current contract execution. Alice can have other contracts with participants other than Bob.

contract may propagate and increase very fast, and each participant with indirect financial interests may have his or her incentive to select a preferred result other than the correct contract execution outcome. When the number of participants who prefer an incorrect result is significantly large, it is very likely that the system will eventually accept the incorrect contract execution result as the final consensus.

Furthermore, participants can always create one or multiple dependent contracts to bribe other participants with financial incentives where they will be rewarded for accepting or outputting a certain preferred contract result deviating from the true execution outcome. Similar threats have been studied by Bonneau in the context of Bitcoin.[5,6]

These two examples show that in certain scenarios, neither Byzantine general fault tolerant[7] nor proof of work with longest chain is applicable as the appropriate model for the blockchain-based smart contract system. They cannot capture characteristics of smart contract participants' behaviors such as runaway judges of contract outcome when they make decisions of contract execution purely according to their own economic interests.

Specifically, participants of the system are rational and may behave as loyal generals or traitors in different situations depending their financial interests and involvements. The number of participants changing their roles can be large enough to affect the assumptions and useful conclusions for blockchain-based smart contract systems derived based on the classic model of Byzantine generals.

In other words, execution results of a smart contract may converge to an outcome that is most profitable to the majority of the participants other than the correct one when participants make decisions regarding contract execution based only on their own economic interests.

Prior Work and Problem Statement

The examples and related discussions in the previous section point to the use of an agent model as the theoretical foundation for blockchain-based smart contracts.

In recent years, there has been an increasing number of studies focusing on bridging game theory and cryptography. Many fundamental cryptographic problems—for example, secret sharing and fair computation—are revisited by adopting a game theoretical point of view that treats all participants as rational players that try to maximize their own profits. Among these, the most relevant are the work of Groce and colleagues[8] and Shareef,[9] which try to introduce rational players into a Byzantine protocol. Groce and colleagues consider the Byzantine problem in which there are loyal generals and a rational adversary who controls a set of traitors. Instead of preventing all participants from reaching a consensus, the adversary tries to maximize his or her own profit. Shareef considers the Byzantine problem with exactly the opposite setting, where there are rational players who want to maximize their own profits and traitors who try to prevent the

participants from reaching a consensus. We remark that the Byzantine protocol is used to reach consensus, yet a blockchain system may well use other protocols, for example, proof of work, for consensus. The influence of rational players in protocols other than Byzantine is much less clear and understood.

Another line of relevant work is the mining game studied by Kiayias and colleagues[10] and Lewenberg and colleagues,[11] which assumes that every miner in the Bitcoin network is a rational player that maximizes his or her own profit by strategically extending a certain branch. All of these prior endeavors are important attempts that strive to incorporate game theory into the study of blockchain-based systems, aiming to better understand the economics and factors affecting participants' decisions behind blockchain.

Following this line of research, we consider a blockchain-based smart contract system, which is much less understood. We assume that all the participants are rational players and discuss their strategic behaviors in smart contract execution. It is apparent that smart contract execution presents a much more complicated scenario than mining. A smart contract may involve multiple players. The economic gain or loss associated with each player may depend on the execution result of the smart contract and may be unknown to any other players in the system. This makes prediction of the behaviors of players much more complicated and prompts a serious question of whether a smart contract can even be correctly executed in the first place.

> **Is it possible for the system to motivate participants to output the correct result using mechanism design, instead of the most profitable choice by the majority?**

Vitalik Buterin, cofounder of one of the most popular smart contract platforms, Ethereum, provided some suggestions along this line of research.[12] Specifically, he conjectured that with a properly designed mechanism of reward and punishment (for example, mandating a deposit and seizing the deposit when the result from a player deviates from the consensus), smart contracts can be correctly executed by players. Does such an incentive or punishment scheme always exist? We will show that this question cannot be answered with a simple yes or no. Indeed, it depends on the behavioral patterns of players in the system.

Can We Guarantee Correct Result as Consensus?

As discussed in previous sections, the classical Byzantine consensus model and proof of work with longest chain cannot fully capture the strategic behaviors of participants in the system. To gain a better understanding of this phenomenon, we take a game theoretical view of smart contract execution. We view the entire smart contract execution as a game, and each participant is a player in the game who tries to maximize his or her own profit.

Taking this viewpoint, let us now revisit Example 1. Suppose participants of the system include Alice, Bob, and others. Each participant needs to decide individually on the coin-flipping result, and the result accepted by a majority of the participants will be adopted by the system as the final consensus. No matter the true flipping result, 0 or 1, and what the other participants decide, Alice will always choose 0. The reason is that by doing so, Alice increases the chance that 0 will be the decision of the majority and the final consensus. Even if it turns out that 1 is accepted by the majority, she loses the same amount of coins as she does if she would have output 1. In game theory, this is called a *dominant strategy*— one that is always no worse than any other strategy regardless of the strategies chosen by other players. We see that Alice's dominant strategy is to decide on 0, while Bob's dominant strategy is to decide on 1. If Alice does not represent a single player but a group of players (for instance, indirectly involved or bribed players) that consists of more than 50 percent of all players, then execution of the smart contract becomes meaningless as the system will always adopt 0 as the coin-flipping result, no matter whether it is true with respect to execution of the contract and coin-flipping algorithm code itself.

A natural question is: Is it possible for the system to motivate participants to output the correct result using mechanism design, instead of the most profitable choice by the majority? A straightforward idea is to introduce reward/punishment, that is, a participant will be penalized by 1 coin if he or she selects the wrong result. Note that in a fully decentralized system, whether a result is correct or wrong depends on consensus of the participants. Therefore, any scheme that tries to reward or punish a participant based on the correctness of his or her decision will essentially become a scheme that rewards or punishes a participant based on whether his or her decision aligns with the majority.

Nevertheless, the simple idea of penalizing a participant if his or her decision is different from the majority

may work in certain cases. We show that, under such a scheme with punishment, always deciding on 0 is no longer the dominant strategy for Alice. This is because in a scenario where every other participant decides on 1, Alice cannot change the result adopted by the system (which is 1) by deciding on 0 alone. Furthermore, she would be penalized if she does so (for instance, her deposit is seized). Hence, the best strategy for Alice now becomes dependent on the strategies of others.

Designing the reward/punishment scheme for players in a game in such a way that a certain economic objective is achieved belongs to the field of mechanism design. For the decentralized model of smart contracts, we aim to design the reward/punishment scheme of participants such that the correct result will be adopted by the system as consensus. We have shown that a simple and straightforward mechanism that penalizes a participant if his or her decision is different from the majority already changes the situation for Alice. Now Alice should first have an estimation of the decisions of other participants, and then make her decision strategically to maximize her own profit. The way that Alice estimates others depends on her behavioral pattern. In the following, we consider common behavioral patterns of participants and discuss whether a mechanism that forces participants to decide on the true result is achievable under these patterns. Which behavioral pattern is the most suitable for participants involved in a smart contract system is an interesting open problem.

Before discussing mechanism design under different behavioral patterns of participants, we first give some definitions. A mechanism for our problem consists of the following:

- a function f_i for each participant i, which takes as input the decision v_j returned by every other participant j, and outputs $f_i(v_1,v_2,\ldots,v_n)$, which is the reward/punishment for participant I; and
- a function g that takes as input the value v_j returned by every participant j, and outputs $v = g(v_1,v_2,\ldots,v_n)$ as the consensus of the system on the outcome of the smart contract.

In addition to the payoff $f_i(v_1,v_2,\ldots,v_n)$, each participant i also has some gains or losses based on the system's decision v, which we define as $g_i(v)$. Notice that g_i is a participant's private information. We assume that every participant is rational in the sense that he or she tries to maximize his or her own payoff, which is $f_i(v_1,v_2,\ldots,v_n) + g_i(v)$. With all these notations, the question becomes whether we can design f_i in such a way that the strategic choices of all participants lead $g(v_1,v_2,\ldots,v_n)$ to be equal to the correct result.

Mechanism Design under Nash Equilibrium

The Nash equilibrium is one of the most common ways to characterize the behavior of players in a game. It studies the state of the game in which no player will gain extra benefits by changing his or her own strategy even if he or she knows the strategies of others. A Nash equilibrium for our problem is the state in which no participant, given the decision of other participants, gains by changing his or her decision. Consider the behavioral pattern of participants such that every participant, knowing the equilibrium strategies of other participants, plays his or her equilibrium strategy. We are concerned with the decision adopted by the system under the equilibrium.

For simplification, we consider only pure strategy for a participant—that is, every participant will deterministically decide on one result, instead of specifying a probability distribution over results. (In an ideal case, a mechanism should ensure that any Nash equilibrium, if it exists, will lead to the correct outcome. That is, if there are multiple Nash equilibria, then under any equilibrium the participants should decide deterministically on the correct result. However, we demonstrate in Theorem 1 that such a mechanism does not exist.)

Theorem 1. There does not exist a mechanism that ensures that, for any Nash equilibrium, g returns the correct outcome.

Proof. Suppose on the contrary that there exists such a mechanism. Consider some smart contract whose outcome is either 0 or 1. There are two possibilities: 0 is the correct outcome, or 1 is the correct outcome. Note that the mechanism does not know the correct outcome in advance, hence it ensures that in both cases, g returns the correct outcome under any Nash equilibrium. However, this is impossible. Assume 0 is the correct outcome and let v_i^* be the output of player i in the Nash equilibrium, then we have $g(v_1^*,v_2^*,\ldots,v_n^*)=0$. Now consider another scenario when 1 is the correct outcome. Because f_i & g_i and g remain the same, $(v_1^*,v_2^*,\ldots,v_n^*)$ is still a Nash equilibrium, and we have $g(v_1^*,v_2^*,\ldots,v_n^*)=0$, which contradicts the fact that g will return the correct outcome in this case.

Given that it is impossible to have a mechanism guaranteeing that participants can decide on the correct result under any Nash equilibrium, a natural question is whether there exists a mechanism guaranteeing that participants can decide on the correct result under some Nash equilibrium. We give a positive answer to this question by Theorem 2.

Theorem 2. There exists a mechanism that ensures there is always a Nash equilibrium such that, under this equilibrium, g returns the correct outcome.

Proof. The design is simple. Let $\{a_1,a_2,\ldots a_m\}$ be all the possible outcomes of the smart contract. We define

g as the function that outputs the majority value—that is, $g(v_1, v_2, \ldots, v_n) = a_{(j^*)}$ where $j^* = \text{argmax}_j \mid v_i \mid v_i = a_j \mid$ (if there are multiple j^*s, choose an arbitrary one). Further, we define f_i such that if $x = (a_j, a_j, \ldots, a_j)$ for any $1 \leq j \leq m$, then $f_i(x) > 0$. Otherwise, $f(x) = 0$. Let a_h be the correct outcome of the smart contract. Consider the situation where every participant's decision is a_h. We claim that this is a Nash equilibrium. To show why, consider an arbitrary participant i. Note that the current payoff for i is $f_i(a_h, a_h, \ldots, a_h) + g_i(a_h)$. If participant i unilaterally changes his or her decision to some $a_{h'} \neq a_h$, we still have $g(a_h, a_h, \ldots a_h, a_h, a_h, \ldots a_h) = a_h$ as g outputs the majority. Consequently, the payoff of participant i changes to $f_i(a_h, a_h, \ldots a_h, a_h, a_h, \ldots a_h) = g_i(a_h)$, which is smaller than $f_i(a_h, a_h, \ldots, a_h) + g_i(a_h)$ by definition. Hence, no participant will unilaterally change his or her decision—that is, every participant decides on the correct result has a Nash equilibrium, and the theorem is proved.

Although Theorem 2 shows that it is possible to design a mechanism that ensures the existence of a Nash equilibrium under which participants reach a consensus on the correct outcome, we remark that such a mechanism cannot rule out the possibility that participants may reach a consensus on the wrong outcome under a different Nash equilibrium. Indeed, no mechanism can rule out such a possibility, as is implied by Theorem 1.

> **We have developed dichotomy theorems given that participants adopt the behavioral pattern of taking their equilibrium strategies.**

Overall, we have developed dichotomy theorems given that participants adopt the behavioral pattern of taking their equilibrium strategies. In the following, we consider other behavioral patterns of participants.

Mechanism Design under Other Behavioral Patterns

Another common behavioral pattern that has received much attention in the literature is super-rationality. A player in a game is considered to have super-rationality if he or she is rational and meanwhile assumes that all other players are super-rational too and that a super-rational individual will always come up with the same strategy as any other super-rational one when facing the same problem.

We revisit Example 1 where all participants are super-rational. Suppose a mechanism using only punishment (without reward) is implemented. Let Alice and Bob represent two groups of participants, A and B. If group A consists of more than 50 percent of participants, then every participant in A will decide on 0. The reason is that every super-rational participant in A faces the same situation: decide on 1 and pay coins to participants in B, or decide on 0 and pay the punishment if the system adopts 1. Because a super-rational participant assumes the same action of others when facing the same situation, every participant in A realizes that deciding on 0 is the more profitable decision. In this case, the system adopts the wrong result if 0 is not correct. This result, as is stated in the following theorem, is already observed in Chen and colleagues' recent paper.[13]

Theorem 3. There does not exist a mechanism with only punishment that ensures g always returns the correct outcome, assuming super-rationality of all the players.

Note that this result rules out only mechanisms that are based solely on punishment. If both reward and punishment are used, then things become more complicated, and it is still open whether a mechanism ensuring the correct outcome is achievable under super-rationality.

Bounded rationality is another kind of behavioral pattern that has received much attention and studies. In our previous discussions, we assume that a player always tries to maximize his or her own profit; however, it is possible that he or she is not aware of the optimal solution or is not able to compute it. In his book *Models of Man*, Simon[14] points out that most people are only partly rational in the sense that they experience limits in formulating and solving complex problems and in processing (receiving, storing, retrieving, and transmitting) information. That means a participant in the system may, for example, not be able to calculate his or her equilibrium strategy, not be able to compute the best strategy even if he or she knows the strategies of others, or simply not be aware of all the strategies available. Bentov, Mizrahi, and Bosenfeld[15] studied the prediction market using blockchain technology and pointed out that it is better to use semi-decentralized model.

We are not able to make universal statements on whether a mechanism that ensures a correct consensus can be achieved, as it crucially relies on the extent of rationality that a participant may possess. In some cases, the simple punishment scheme that penalizes a participant if his or her decision is different from majority may work: If a participant knows very limited information of other participants, then he or she may rely only on the past data in the smart contract system

to make a decision. If he or she observes that most of the past smart contracts are correctly executed, then this participant is likely to decide on the correct result. If most of the participants are like him or her, then other participants that are more strategically powerful would also follow the correct result. The above reasoning is plausible. However, a formal argument, clearly specifying the conditions under which a mechanism ensures the correct execution of smart contracts, is desired. This is an important open problem. ■

References

1. G. Karame and E. Audroulaki, *Bitcoin and Blockchain Security*, Artech House, 2016.
2. A. Gervais et al., "On the Security and Performance of Proof of Work Blockchains," *CCS*, 2016, pp. 3–16.
3. L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, 1982, pp. 382–401.
4. M. Vukolić, "The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication," *International Workshop on Open Problems in Network Security*, Springer, 2015, pp. 112–125.
5. J. Bonneau, "Why Buy When You Can Rent?" *FC*, 2016, pp. 19–26.
6. J. Bonneau, J. Clark, and S. Goldfeder, "On Bitcoin as a Public Randomness Source," IACR Cryptology ePrint Archive, vol. 2015, 2015, p. 1015.
7. A. Groce et al., "Byzantine Agreement with a Rational Adversary," *ICALP*, 2012, pp. 561–572.
8. A. Groce and J. Katz, "Fair Computation with Rational Players," *CRYPTO*, 2012, pp. 81–98.
9. A. Shareef, "Distributed Rational Consensus," IACR Cryptology ePrint Archive, vol. 2010, 2010, p. 330.
10. A. Kiayias et al., "Blockchain Mining Games," *EC*, 2016, pp. 365–382.
11. Y. Lewenberg et al., "Bitcoin Mining Pools: A Cooperative Game Theoretic Analysis," *AAMAS*, 2015, pp. 919–927.
12. V. Buterin, "Blockchain and Smart Contract Mechanism Design Challenges," *FC Workshop on Trusted Smart Contracts*, 2017.
13. L. Chen et al., "Decentralized Execution of Smart Contracts: Agent Model Perspective and Its Implications," *WTSC*, 2017.
14. H.A. Simon, *Models of Man: Social and Rational*, Wiley 1957.
15. I. Bentov, A. Mizrahi, and M. Rosenfeld, "Decentralized Prediction Market without Arbiters," arXiv preprint arXiv:1701.08421, 2017.

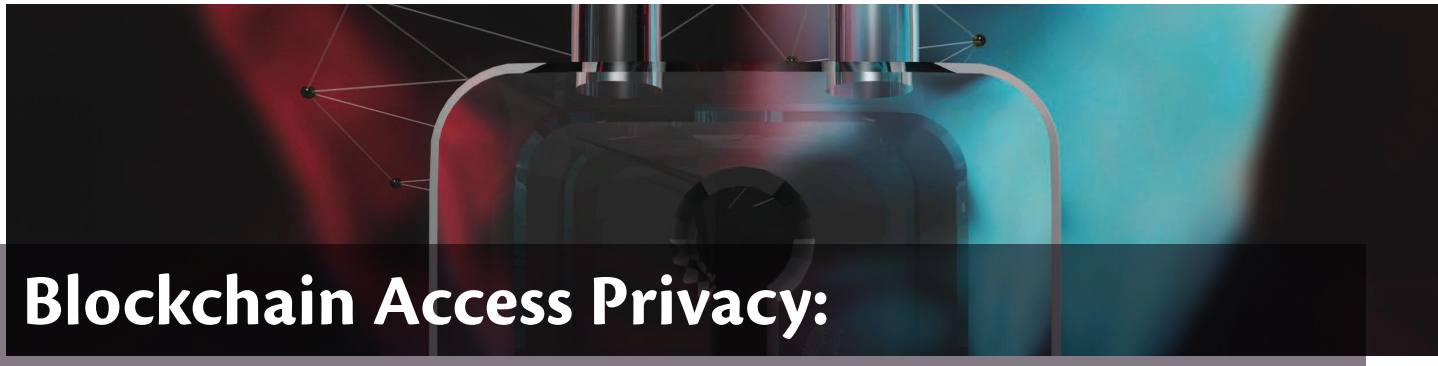
Lin Chen received a PhD in computer science from Zhejiang University, China, in 2013. From 2013 to 2016, he worked as a postdoctoral fellow at the Technical University of Berlin and then the Hungarian Academy of Science. He is currently a research assistant professor at the University of Houston. His research interests include blockchain, stochastic optimization, parameterized algorithms, and complexity. Email him at chenlin198662@gmail.com.

Lei Xu received a PhD in computer science from the Institute of Software, Chinese Academy of Sciences, in 2011. He is currently research assistant professor at the University of Houston. From 2011 to 2013, he worked as a research engineer at the Central Research Institute, Huawei Technologies. His research interests include blockchain, cloud security, and applied cryptography. Email him at xuleimath@gmail.com.

Zhimin Gao is a PhD candidate of computer science at the University of Houston. He is currently working as a research assistant at the University of Houston. His research interests include blockchain, high-performance computing, and cloud computing. Email him at mtion@msn.com.

Yang Lu received an MS in information technology from Southern Polytechnic State University. Her research is focused on computing security and blockchain. She previously worked as a software engineer in FileVison. Currently, she works as project manager in the Department of Computer Science at the University of Houston. Email her at ylu17@central.uh.edu.

Weidong (Larry) Shi received a PhD in computer science from the Georgia Institute of Technology where he did research in computer architecture and computer systems. He was previously a senior research staff engineer at Motorola Research Lab, Nokia Research Center and co-founder of a technology startup. Currently, he is employed as an associate professor by the University of Houston. Email him at wshi3@central.uh.edu.



Blockchain Access Privacy:

Challenges and Directions

Ryan Henry | University of Calgary
Amir Herzberg | University of Connecticut
Aniket Kate | Purdue University

Most blockchain users remain susceptible to privacy attacks. Many researchers advocate using anonymous communications networks, such as Tor, to ensure access privacy. We challenge this approach, showing the need for mechanisms through which non-anonymous users can publish and fetch transactions without enabling others to link those transactions to their network addresses or to their other transactions.

A blockchain is a distributed, append-only log of time-stamped records that is cryptographically protected from tampering and revision. In the eight years since blockchains were first proposed, their use as publicly accessible and verifiable ledgers for online financial transactions has become widespread. This rapid adoption has largely been spurred by the success of Bitcoin (https://www.bitcoin.org), a digital currency that—owing to its decentralized and pseudonymous nature, support for complex financial instruments (enabled by a powerful, built-in scripting language), and capacity to facilitate fast and inexpensive transactions across the globe—has proven to be a highly disruptive force in the finance and e-commerce sectors.

As Bitcoin and alternatives like Ethereum (https://www.ethereum.org) and Ripple (https://ripple.com) continue to mature and grow in market value, it is becoming increasingly likely that blockchains as a means to facilitate financial transactions are here to stay. Yet blockchains represent far more than a mere monetary innovation; researchers and industry members alike are only just beginning to understand the true potential of blockchain-based distributed ledgers, with their strong integrity and availability guarantees and their ability to leverage community consensus to eschew centralized trusted curation. Indeed, beyond the sorts of payment transactions for which blockchains are already widely deployed, potential applications for blockchains abound in areas as diverse as electronic voting, certificate authorities, the Internet of Things, and smart systems. Moreover, the past few years were marked by announcements from numerous companies—ranging from startups like R3 (https://www.r3.com) to established technology firms like IBM and financial institutions like Visa—about forthcoming products based on innovative blockchain designs that are specially tailored to meet organizational and business logic needs. The target applications for these products range from payment settlement through supply-chain management and beyond.

Just how private are today's blockchains? The ephemeral nature of users' pseudonymous identities in Bitcoin played a key role in its early success. However, eight years of intense scrutiny by privacy researchers has

 Copublished by the IEEE Computer and Reliability Societies

brought to bear an arsenal of powerful heuristics using which attackers can effectively link disparate Bitcoin transactions to a common user and, in many cases, to that user's real-world identity. Ultimately, instead of providing the bastion of privacy for financial transactions that its early adopters envisioned, Bitcoin and its altcoin brethren are in many ways less private than traditional banking, where government regulations mandate basic privacy protections. In an attempt to address this situation, the cryptography and privacy research communities have proposed and implemented several protocols aiming to improve blockchain privacy. These protocols all try to decouple users' pseudonymous identities from the specific transactions they make, thereby frustrating attempts to link transacting parties based on data that appears in the blockchain. However, none of the proposed protocols attempts to hide the identities of users from network-level adversaries as the users publish or retrieve data from the blockchain. Instead, the proposed protocols "outsource" this crucial step, relying on an external anonymous communications network such as Tor (https://www.torproject.org). However, running complex protocols over general-purpose, low-latency anonymity networks such as Tor is fraught with risks and can expose users to subtle-yet-devastating deanonymization attacks, thereby undermining the privacy guarantees of the entire blockchain system. We can do better!

Cryptography to the Rescue?

Most blockchains are, at their core, massively distributed and publicly accessible databases; therefore, beyond ensuring that the data they store does not, in and of itself, betray user privacy, any research program that seeks to fully address blockchain privacy must additionally consider (at the very least) privacy for two fundamental types of transactions: reading data from and writing data to a blockchain.

In the context of cryptocurrencies like Bitcoin, the database represented by the blockchain is a publicly accessible and verifiable ledger of financial transactions. Specifically, whenever a transaction occurs, the originating party publicly announces the transaction to a handful of selected entities, who then spread the details of that transaction throughout the network via a gossip protocol. The transaction is ultimately aggregated with several other (unrelated) transactions into a discrete block, which then gets irreversibly appended to a chain comprising all earlier blocks. The chain of blocks can—indeed, to obtain strong integrity and availability, must—be replicated and shared in its entirety among many nodes in a network, thereby providing each node with a global, eventually consistent view of every transaction that has ever taken place. New transactions are reflected in all replicas of the blockchain within some predefined expected time, which can range from a few seconds (for instance, in Ripple) to a few minutes (for instance, in Bitcoin).

Each transaction is associated with a pair of pseudonyms (often called *wallets*), respectively identifying the sender and receiver of some digital assets. Users can generate new pseudonymous wallets with which to receive digital assets arbitrarily and at will; it is considered a best practice for Bitcoin users to generate a fresh, ephemeral wallet whenever they wish to conduct a new transaction. The primary motivation for generating such ephemeral wallets is to protect user privacy by making it difficult for an attacker to link together the various transactions involving a given user by simply examining the sender and receiver pseudonyms appearing in transactions recorded in the ledger. However, as Bitcoin and related altcoins grow ever-more prevalent, there is a growing concern that the "privacy" offered by this approach is illusory at best. Indeed, as mentioned previously, the past eight years of research into blockchain privacy has given rise to a veritable treasure trove of effective heuristics using which attackers can link Bitcoin transactions back to a common user, despite the widespread use of ephemeral wallets.[1–3]

Figure 1 depicts a traditional blockchain architecture. (We use the qualifier "traditional" here to differentiate the blockchain architectures we consider from those involving payment channels and other layer-2 applications, which introduce a host of new privacy concerns that are beyond the scope of this article.) For the purposes of this article, we focus on the two arrows that are bold and highlighted in green; specifically, we focus on the need for innovative mechanisms that allow users to

- announce and publish transactions anonymously, a task for which we envision a tailor-made anonymity mechanism that is integrated directly into the blockchain architecture; and
- fetch transactions privately, a task for which we envision using special private information retrieval (PIR) protocols designed and optimized to support efficient and expressive queries for transactions stored in a blockchain.

We note that a handful of second-generation altcoins—including Zcash (https://z.cash) and Monero (https://getmoreno.org)—natively employ cryptographic techniques to prevent the contents of transaction on the blockchain from leaking private information about transacting parties. Likewise, the research literature contains several proposals (a selection of which we summarize in the next subsection) that aim to

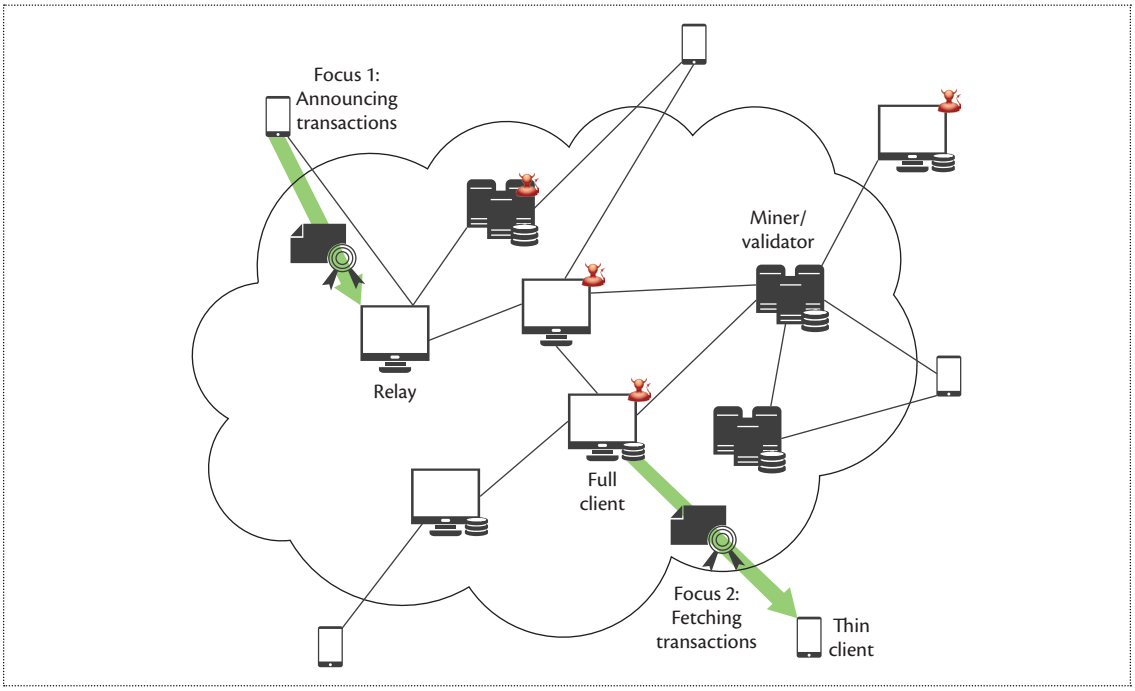


Figure 1. Topology of a typical blockchain system. The two bold arrows (highlighted in green) illustrate sensitive information flows that must be protected to prevent attackers from leveraging network-level information to compromise the privacy of blockchain users.

provide similar transaction privacy atop the deployed Bitcoin, Ripple, and Ethereum blockchains. While such approaches are effective at protecting blockchain users against a subset of the deanonymization heuristics that plague mainstream deployed blockchains, we emphasize that the existing approaches, so far, focus on preventing the data stored in a blockchain from leaking private information—they do nothing significant to mitigate against inferences that leverage network-level information (for example, IP addresses) or access patterns (for example, specific blocks or portions thereof) revealed when users interact with the blockchain data. As such, the existing proposals all fall far short of solving the blockchain privacy problem in its entirety.

Existing Protocols for Transaction Privacy

As the insufficiency of ephemeral pseudonyms became apparent to the Bitcoin community, a proposal called CoinJoin emerged as a potential solution. In CoinJoin, users route their transactions through a centralized mixing service (sometimes called a tumbler), which serves to obscure the relationships between the senders and receivers of those transactions before they are posted to the ledger. However, such centralized mixing services introduce a single point of trust and failure; indeed, the mixing service always knows the link between the sender and receiver of each transaction, and perhaps more troublingly, there is nothing to stop the mixing service from stealing assets that users try to route through it. A series of progressively more sophisticated protocols have been proposed to address CoinJoin's limitations.

The first improvement was Mixcoin, which attempts to mitigate the risk of theft by holding the mixing service "accountable" if it steals a user's assets (though theft is still technically possible and the mixing service still learns who is transacting with whom). Building on a series of incremental improvements to this basic idea (including BlindCoin and Blindly Signed Contracts), a proposal called TumbleBit[4] finally addressed the accountability and anonymity weaknesses of Mixcoin in a manner fully compatible with Bitcoin; however, the TumbleBit approach requires upwards of 20 minutes (that is, two Bitcoin blocks) per transaction on average and introduces additional transaction fees. Aniket Kate's CoinShuffle and CoinShuffle++[5] take a different approach, having users perform a special multiparty computation among themselves so that no third-party mixing service is necessary.

The emerging privacy-centric cryptocurrencies, such as Zcash and Monero, employ cryptographic primitives such as zero-knowledge succinct noninteractive argument of knowledge (zk-SNARK), traceable ring signatures, confidential transactions, and stealth addresses to offer significantly better privacy properties than those possible for Bitcoin transactions.

Inadequacy of Existing Proposals

The above transaction privacy protocols all aim to sever the link between senders and receivers as recorded in the transactions that get published to the blockchain. However, the approaches are all susceptible to attacks that reestablish links between transacting parties using network-level information and/or access patterns observed both as users announce their transactions and as they probe the blockchain to learn which of their transactions have posted to the ledger.[2] For example, an attacker who observes that a given user visits a website immediately before that website receives a donation via Zcash or Monero might surmise that the user made the donation; moreover, the attacker can all but confirm this suspicion if it later observes the same user checking whether the transaction in question has posted to the ledger.

To define away this elephant in the room, the developers of such privacy protocols typically assume that users communicate over an anonymous communication protocol such as Tor; in fact, some privacy-centric altcoins—like Zcash, Anoncoin, and Torcoin—include native support for Tor and expect that all users interact with their blockchains exclusively through Tor. As an example, the Zcash website clearly states (and we quote) that "a unique IP address can allow network observers to correlate your Zcash transactions with each other and with your other traffic" to which it adds that "advanced users may opt to connect through Tor to obfuscate their node's IP address, however, further exploration is needed on a vulnerability combining Bitcoin's Denial of Service mitigation (inherited into Zcash) and anonymous communication networks like Tor before we can recommend users who are not familiar with the attack to route their Zcash nodes through Tor" (https://z.cash/support/security/privacy-securityrecommendations.html).

This dependency on Tor for anonymity introduces some rarely acknowledged yet undeniably troubling weaknesses. One source of weakness stems from the fact that Tor is specifically designed to support low-latency communication, such as interactive web browsing and real-time instant messaging; indeed, it seems inherent (and real-world attacks seem to confirm) that such low-latency low-bandwidth anonymous communication systems can provide at most a relatively weak form of anonymity compared to high-latency approaches like Chaumian mix networks or high-bandwidth approaches like dining-cryptographers (DC) networks. Indeed, a recent paper by Das and colleagues[6] analyzed the so-called "anonymity trilemma" and concluded that, in the presence of a global passive (network-level) adversary, anonymous communications networks can hope to provide just two of three desirable properties: strong anonymity, low bandwidth overhead, and low latency overhead. Fortunately, because financial transactions are naturally able to tolerate moderate latency—so-called "permissionless blockchains," like the one used in Bitcoin, already impose latencies on the order of several minutes even without the use of an anonymous communications network—users need not settle for the relatively weak anonymity guarantees that low-latency systems like Tor can provide.

Further, Biryukov and Pustogarov[7] demonstrated how Bitcoin's "blacklisting" measures may ultimately leave users conducting Bitcoin transactions over Tor more vulnerable to active deanonymization attacks than those announcing their transaction nonanonymously. They describe man-in-the-middle attacks that exploit the Bitcoin network's built-in reputation-based denial-of-service protection mechanism to force specific Bitcoin peers to ban Tor exit relays of the attacker's choice, thus forcing all Bitcoin traffic to exit the Tor network through a small set of attacker-controlled relays. Once in this privileged position, the attacker can launch several troubling privacy attacks, including deanonymization via traffic correlation (which is made easier because the attacker automatically controls one end of the communication), correlating multiple wallet addresses to a common user, and launching "double-spending" attacks by lying to thin clients about previous transactions involving a given wallet address.

Yet another problem arises from the fact that Tor is often blocked by IT departments within organizations or even subject to state-level censorship by authoritarian governments. This has direct negative consequences for the privacy of users connecting from such organizations or countries, even though the censorship is almost certainly intended to quell some other, unrelated usage of Tor. As a workaround for such censorship, Tor ships with support for some censorship-evasion techniques including Tor bridges and pluggable transports; however, the effectiveness of these mechanisms is far from perfect and censorship events continue to affect Tor users. In general, it seems unwise to advocate the wholesale use of censorship circumvention tools for activities that are typically not subject to censorship.

Moreover, a third-party anonymous communication network such as Tor may not be willing or able to support blockchain traffic on a large scale. A dual concern is some blockchain systems may be hesitant to use Tor since Tor has been used for nefarious purposes, ranging from ransomware and botnet command and control to child pornography. As an anecdotal example of this, Kate has learned through communications with developers at Ripple that, despite being very keen on improving privacy for their clients, Ripple's developers are unwilling to leverage a Tor-based solution to do so.

Finally, due to their decentralized design, blockchain systems seem like prime candidates for fulfilling their own anonymity and privacy needs, avoiding the dependency on external services and providing performance and privacy/anonymity guarantees tailored to their own needs.

In short, we believe that effective blockchain privacy necessitates rethinking the one-size-fits-all approach of using external anonymous communications infrastructures to solve all problems requiring anonymity. Although anonymity does indeed love company, mixing two dissimilar types of traffic together does not necessarily improve anonymity for either type and, if not done very carefully and correctly, may in fact provide weaker anonymity than protecting each type of traffic with its own tailor-made solution.

Publishing Transactions Anonymously

By their very design, blockchain systems require extensive overlay networks through which participants announce transactions and agree on what transactions should ultimately appear on the blockchain. Thus, it seems natural to leverage the existing overlay structure to realize anonymous transaction publishing, rather than relying on an external service like Tor. We propose that blockchain privacy protocols should de-link users' network-level information from their transactions using mechanisms that piggyback on the overlay network that is already in place for announcing transactions. The specifics of how such a mechanism might work vary, depending on the structure of the overlay network imposed by the consensus protocol—that is, depending on how participants decide which transactions qualify for inclusion in the blockchain.

Proposed and deployed blockchains fall into two distinct categories based on the mechanism they use to build a consensus around what data to immortalize in the blockchain: *permissionless* and *permissioned*.

The blockchains underlying Bitcoin and Ethereum constitute two prominent examples of permissionless blockchains. As their name implies, permissionless blockchains place no restrictions on who participates in the consensus process. Instead, unrestricted entities called miners collectively decide which blocks should be appended to the chain by providing an associated proof of work. In the case of Bitcoin, this proof of work takes the form of a "partial hash inversion," wherein the miners seek inputs that lead a cryptographic hash function to produce a digest whose numerical value does not exceed some global-parameter target.

Such a permissionless consensus guarantees that only valid blocks get appended to the blockchain (approximately) under the assumption that more than half of all mining resources in the network are controlled by honest—or, at least, noncolluding—entities.

The blockchains underlying Ripple and the Linux Foundation's Hyperledger (https://www.hyperledger.org) are two prominent examples of permissioned blockchains. In contrast to permissionless blockchains, permissioned blockchains do place restrictions on who participates in the consensus process. A group of highly available entities (with strong identities) collectively decide which blocks should be appended to the chain by leveraging a Byzantine fault-tolerant atomic broadcast protocol. This approach allows permissioned blockchains to reach consensus very rapidly, requiring as little as a few seconds for each transaction to be reflected in the ledger.

The contrasting security assumptions and efficiency guarantees of permissionless and permissioned blockchains make them well suited to different use cases, and indeed, the two varieties are prospering together: traditionally structured organizations/consortiums are increasingly adopting permissioned blockchains, while peer-to-peer (P2P) solutions continue to leverage permissionless blockchains.

Publishing to Permissionless Blockchains

Permissionless blockchain systems (like Bitcoin and Ethereum) employ P2P networks of relays to propagate transactions and blockchain updates throughout the network using a best-effort gossip protocol. Such P2P networks typically experience considerable churn, with relays joining, leaving, and rejoining the network at will; however, the average number of relays in the network at any given time can remain relatively high. For example, at the time of writing, the number of online relays in the Bitcoin network at any given time is about one-and-a-half times the number of Tor relays. (As of 4 October 2017, Tor Metrics estimates about 6,300 Tor relays [https://metrics.torproject.org] versus the Bitnode estimate of about 9,900 full Bitcoin nodes [https://bitnodes.21.co].) One might, therefore, consider employing the elaborate Bitcoin communication infrastructure toward improving the anonymity of users' announcements. Given the P2P nature of the network, we believe it may be possible to leverage the existing academic research on P2P anonymous communications networks. For instance, such a solution could be based on Pisces,[8] employing the social trust links to construct anonymous communication paths that are robust to compromise in the presence of route-capture attacks and Sybil nodes. However, given the dynamic and open nature of permissionless blockchains such as Bitcoin, establishing trust in relays will be a prominent challenge.

The Kovri project (https://www.getcorvi.org), an offshoot of the Monero and Bitcoin developers' recent interest in the Dandelion networking policies,[9] clearly

indicates the blockchain community's awareness of the problem; nevertheless, significant efforts are necessary going forward. In general, it will be an interesting challenge to analyze and establish security, privacy, and viability of P2P anonymous communications system over permissionless blockchain systems.

Publishing to Permissioned Blockchains

Permissioned blockchain systems (like Ripple, Corda [https://www.corda.net], and Hyperledger) employ a clique of highly available validator nodes for agreeing on transactions and blocks. These nodes employ traditional asynchronous Byzantine-tolerant consensus protocols to append a block of transactions to the blockchain. Here, validators select valid transactions to be agreed on from those transactions forwarded by system users. As typically transactions from several users are added to any given block, a simple approach to provide anonymity here will be to perform all the communication between users and validators over an anonymous communications network. However, we advocate improving efficiency and reducing the overhead by combining the consensus process for agreeing on transactions with the process of mixing users' announcements.

This can be modeled as an asynchronous multiparty computation (AMPC) problem and can be solved using the generic AMPC techniques; however, we propose development of tailored solutions to further improve the efficiency. A possible tailored approach for agreeing on a randomly permuted set of transactions can involve combining Newton's identity method for power sums (as employed by Ruffing and colleagues[5]) with asynchronous verifiable secret sharing and asynchronous Byzantine consensus. Nevertheless, a key challenge will be to make these solutions scale well (possibly sublinearly) with the number of mixed transactions.

Fetching Transactions Privately

Blockchains differ from traditional databases in their use of cryptography as a means to eschew both centralization and trusted curators, all the while ensuring strong resistance to "tampering" (that is, history rewriting). Yet this remarkable combination of attributes is guaranteed only for users that hold a complete local replica of the blockchain. With a blockchain currently over 100 GB and growing, this local-storage requirement is quickly becoming infeasible for casual Bitcoin users; as a result, many such users now employ so-called *thin clients* that bypass the need to hold a local copy of the blockchain by forwarding blockchain queries to semi-trusted intermediaries.

Specifically, thin clients run in what is called Simplified Payment Verification (SPV) mode—so named after the section of the original Bitcoin white paper[10] that details it—wherein the initial syncing process connects to an arbitrary full node and downloads only the block headers (each of which includes a Merkle root committing to the actual block). The thin client then verifies that the given headers indeed form a blockchain (with sufficient difficulty value), after which they can request the details of transactions matching certain patterns (for example, payments to or from particular addresses) from any full node. The full nodes reply to such requests with a copy of any relevant transactions together with Merkle branches linking those transactions to their associated block headers. This process exploits the Merkle tree structure to allow proofs of inclusion in a block without needing to provide the thin client with the full contents of the block.

The SPV approach has the distinct advantage that the cost of initial syncing scales linearly with the length of the blockchain (about 80 bytes per header, or 4.2 MB per year) and is independent of the size of the actual blocks. However, a naive implementation of SPV exposes thin clients to potentially devastating attacks on privacy. As a thin client will typically request details about precisely those transactions that correspond to keys it owns, it may end up revealing to the full node a complete list of its public addresses. In particular, Bitcoin users that rely on such thin clients are subject to deanonymization. This is a serious risk; there have been numerous reports of high-rolling Bitcoin users being identified and targeted by miscreants to steal their digital fortunes (https://bitcointalk.org/index.php?topic=16457.0).

A tempting response is to route thin-client queries through an anonymity network like Tor; however, this leaves clients susceptible to low-cost deanonymization and double-spending attacks.[7] Indeed, the root problem for thin clients is not a lack of anonymity for the querier but, rather, a lack of privacy for the queries—anonymity, quite simply, solves the wrong problem. Instead, we observe that the problem of realizing private blockchain queries is imminently solvable using a well-known cryptographic primitive called private information retrieval.

PIR is a cryptographic primitive that solves the seemingly impossible problem of letting clients query a remote database, while not exposing the clients' query terms or the responses they generate to the database operator. PIR has received considerable attention from the cryptography, privacy, and theoretical computer science research communities. Alas, despite a series of significant advances over the past two decades, existing PIR techniques are notoriously inefficient, and consequently, to date not one of the numerous PIR-based applications proposed in the research literature has been deployed at scale to protect the privacy of users "in the wild."

As a result, transitioning the idea of using PIR to fetch blockchain transactions privately into practice still necessitates some basic research and rather substantial engineering and implementation efforts. Fortunately, some recent advances in PIR research yield the promise of PIR protocols that are sufficiently practical to deploy on databases of size commensurate with Bitcoin's blockchain.

Private Blockchain Queries from PIR

The key goals here are to create protocols that enable thin clients to

- determine if particular transactions are reflected in the blockchain (and, if so, how many blocks have been appended since, a rough proxy for the computational effort that would be required to "undo" that transaction); and
- find out the balances associated with a set of public keys, reflecting all transactions that have occurred so far involving those keys.

This will involve defining appropriate data structures that lend themselves to being queried via PIR as well as efficient mechanisms for keeping those data structures up to date as the blockchain grows. Although one could conceptually employ any PIR protocol for this purpose, thinking toward mass adoption among the millions of present and potential Bitcoin users, we suggest very strict requirements on acceptable communication and computation overhead. In effect, the target will be communication costs that are reasonable for a smartphone communicating over a mobile data connection, and computation costs low enough for a modestly equipped server to process tens or hundreds of queries every second. Such strict requirements preclude most existing PIR protocols; however, the recent introductions of distributed point functions,[11] Intel's Software Guard Extensions (SGX) architecture (https://software.intel.com/sgx), and Ryan Henry's indexes of queries[12] provide three very elegant—and, we believe, highly practical—ways to realize the kinds of PIR-based private blockchain queries we envision. Each approach brings its own performance characteristics and its own security assumptions, ranging from noncollusion to computational assumptions to trusted hardware. The research objective here will be to devise appropriate data structures to facilitate PIR-based queries over blockchain data, and then to implement and evaluate the suitability of the various approaches.

Moreover, by leveraging the anonymous communications framework we advocated earlier, it may be possible to realize lower-cost relaxations of information-theoretic PIR that satisfies a differentially private notion for private queries.[13]

General-purpose anonymous communications systems like Tor are not a panacea for communication privacy issues. Indeed, not all applications are anonymized equally well by low-latency anonymity networks, and not all privacy problems are adequately addressed by making users anonymous.

While we only considered ways to address privacy challenges arising from network-level and access pattern leakage on traditional blockchains, new blockchain extensions—such as the lightning network (https://lightning.network), which has been recently proposed as a way to greatly improve the scalability of permissionless blockchains—introduce new subtle privacy challenges that will also require novel solutions. Although some solutions are already emerging to improve privacy in these path-based transactions,[14,15] it is an interesting open challenge to devise scalable mechanisms for performing (multihop) payment-channel transactions privately against a network-level adversary. ■

Acknowledgments

This material is based on work supported by the National Science Foundation under grants 1718595 and 1719196, and by United States-Israel Binational Science Foundation (BSF) under grant 2016718.

References

1. M. Meiklejohn et al., "A Fistful of Bitcoins: Characterizing Payments among Men with No Names," *Communications of the ACM*, vol. 59, no. 4, 2016, pp. 86–93; https://doi.org/10.1145/2896384.
2. P. Koshy, D. Koshy, and P.D. McDaniel, "An Analysis of Anonymity in Bitcoin Using P2P Network Traffic," *Proceedings of FC*, LNCS 8437, 2014, pp. 469–485; https://doi.org/10.1007/978-3-662-45472-5_30.
3. E. Androulaki et al., "Evaluating User Privacy in Bitcoin," *Proceedings of FC*, LNCS 7859, April 2013, pp. 34–51; https://doi.org/10.1007/978-3-642-39884-1_4.
4. E. Heilman et al., "TumbleBit: An Untrusted Bitcoin-Compatible Anonymous Payment Hub," *Proceedings of NDSS*, 2017; https://doi.org/10.14722/ndss.2017.23086.
5. T. Ruffing, P. Moreno-Sanchez, and A. Kate, "P2P Mixing and Unlinkable Bitcoin Transactions," *Proceedings of NDSS*, 2017; https://doi.org/10.14722/ndss.2017.23415.
6. D. Das et al., "Anonymity Trilemma: Strong Anonymity, Low Bandwidth Overhead, Low Latency—Choose Two," *Proceedings of IEEE S&P*, 2018; https://eprint.iacr.org/2017/954.
7. A. Biryukov and I. Pustogarov, "Bitcoin over Tor Isn't a Good Idea," *Proceedings of IEEE S&P*, 2015, pp.122–134; https://doi.org/10.1109/sp.2015.15.
8. P. Mittal, M.K. Wright, and N. Borisov, "Pisces: Anonymous Communication Using Social Networks,"

Proceedings of NDSS, 2013; https://arxiv.org/abs/1208.6326.
9. S. Bojja Venkatakrishnan, G. Fanti, and P. Viswanath, "Dandelion: Redesigning the Bitcoin Network for Anonymity," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 1, pp. 22:1–22:34; http://doi.acm.org/10.1145/3084459.
10. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Nov. 2008; https://bitcoin.org/bitcoin.pdf.
11. N. Gilboa and Y. Ishai, "Distributed Point Functions and Their Applications," *Proceedings of EUROCRYPT*, LNCS 8441, 2014, pp. 640–658; https://doi.org/10.1007/978-3-642-55220-5_35.
12. S.M. Hafiz and R. Henry, "Querying for Queries: Indexes of Queries for Efficient and Expressive IT-PIR," *Proceedings of CCS*, 2017, pp. 1361–1373; https://doi.org/10.1145/3133956.3134008.
13. R.R. Toledo, G. Danezis, and I. Goldberg, "Lower-Cost ϵ-private Information Retrieval," *Proceedings on Privacy Enhancing Technologies* (PoPETs 16), vol. 2016(4), 2016, pp. 184–201; https://doi.org/10.1515/popets-2016-0035.
14. G. Malavolta et al., "Concurrency and Privacy with Payment-Channel Networks," *Proceedings of CCS*, 2017; https://doi.org/10.1145/3133956.3134096.
15. M. Green and I. Miers, "Bolt: Anonymous Payment Channels for Decentralized Currencies," *Proceedings of CCS*, 2017, pp. 473–489; http://doi.acm.org/10.1145/3133956.3134093.

Ryan Henry is an assistant professor in the Department of Computer Science at the University of Calgary. His focus is on the systems challenges of applied cryptography, with an emphasis on using cryptography to build secure systems that protect the privacy of their users. Contact him at ryan.henry@ucalgary.ca.

Amir Herzberg is a Comcast professor for security innovation at the Department of Computer Science and Engineering, University of Connecticut. Previously he was with Bar Ilan University and with IBM Research. His research areas include network and web security, applied cryptography, privacy and anonymity, usable security, and security for cyber-physical systems. Contact him at amir.herzberg@uconn.edu.

Aniket Kate is an assistant professor in the Computer Science Department at Purdue University. He designs, implements, and analyzes privacy and transparency enhancing technologies. Contact him at aniket@purdue.edu.

When the "Crypto" in Cryptocurrencies Breaks: Bitcoin Security under Broken Primitives

Ilias Giechaskiel, Cas Cremers, and Kasper B. Rasmussen | University of Oxford

We present a systematic analysis of the ways in which Bitcoin's core cryptographic building blocks can break and the subsequent effect on Bitcoin security guarantees. Our analysis reveals effects ranging from minor privacy violations to a complete breakdown of the currency and offers insights for other cryptocurrencies and the Bitcoin migration plans.

Cryptocurrencies such as Bitcoin rely on cryptographic primitives for their guarantees and correct operation. Such primitives typically weaken over time, due to progress in cryptanalysis and advances in the computational power of the attackers. It is therefore prudent to expect that, in time, the cryptographic primitives used by Bitcoin will be partially, if not completely, broken.

In anticipation of such breakage, the Bitcoin community has created a wiki page that contains draft contingency plans.[1] However, these plans are informal and incomplete: no adequate transition mechanism has been built into Bitcoin, and no plans for weakened primitives have been considered. Primitives rarely break abruptly: for hash functions, it is common that first a single collision is found. This is then generalized to multiple collisions, and only later do arbitrary collisions become feasible to compute. In parallel, the complexity of attacks decreases to less than brute-force, and the computational power of attackers increases.

Even if such attacks are years away from being practical, it is crucial to anticipate the impact of broken primitives so that appropriate contingency plans can be put in place. Our work contributes to filling this gap. We provide the first systematic analysis of the impact of broken primitives on Bitcoin. By analyzing the failure of primitive properties, both in isolation and in combination, we describe the range of consequences different breaks have and pinpoint their exact cause.

We show that a break in RIPEMD160 can allow an attacker to repudiate payments. SHA256 collisions and second pre-image attacks, as well as selective forgery of Elliptic Curve Digital Signature Algorithm (ECDSA) signatures, all allow an adversary to steal or destroy coins. Finally, SHA256 pre-image attacks have the most severe consequences, as they allow an attacker to take complete control over the Bitcoin system, but only through exploiting the flexibility of the coinbase transaction.

Our investigations raise concerns about the currently specified migration plans for Bitcoin, being overly conservative in some respects and inadequate in others. To that end, we make suggestions regarding future

 Copublished by the IEEE Computer and Reliability Societies

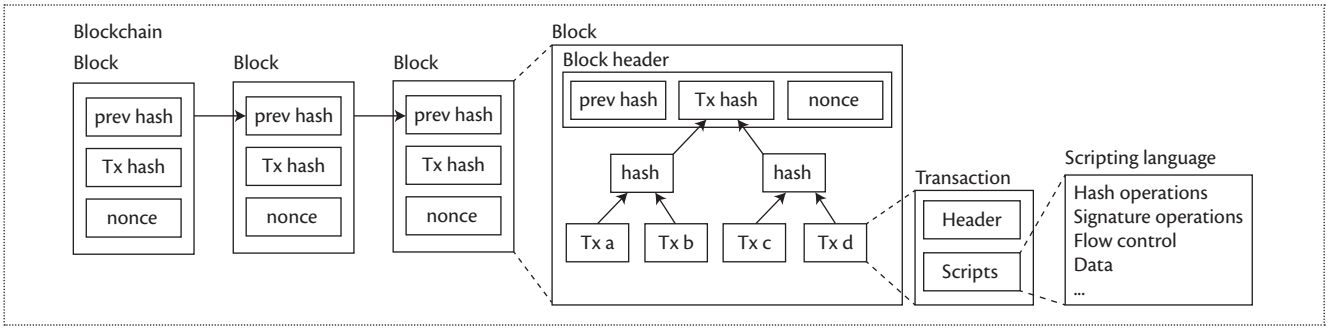


Figure 1. The blockchain data structure. This forms the basis of the public, append-only ledger, where all transactions are recorded.

iterations of Bitcoin in response to entirely broken and partially weakened primitives and relate Bitcoin's security model to that of other currencies.

Background

Bitcoin is a popular peer-to-peer (P2P) cryptocurrency introduced in 2008 by Satoshi Nakamoto.[2] Figure 1 shows a high-level view of the main component of Bitcoin—the blockchain—which will guide this section. The blockchain is a public log of all Bitcoin transactions that have occurred, combined together in components called blocks. Transactions use a scripting language that determines the owners of coins, and it is up to "miners" to verify that only valid transactions occur. To ensure that nobody can change or remove past transactions, miners have to solve a hard computational puzzle, known as a proof of work (PoW). The final component of Bitcoin is its underlying P2P network, which enables distributed communication.

Transactions and Scripts

Bitcoin is an electronic cash system,[2] so *transactions* to transfer coins between users are central to its structure. A transaction is a list of inputs—unspent transactions in the blockchain—and a list of outputs—addresses to which to transfer the coins, whose unit is a "satoshi," equal to 10^{-8} bitcoins or BTCs. To ensure that only the owner can spend his or her coins, each input and output is accompanied by a *script*. For outputs, this "locking" script contains the conditions under which the output can be redeemed (`scriptPubKey`), while for inputs, an "unlocking" script contains a cryptographic signature (`scriptSig`) as proof that these conditions have been met. These scripts are sequences of instructions (`opcodes`) that get executed by miners. To prevent denial-of-service (DoS) attacks exploiting computationally intensive instructions, most nodes accept only the *five standard scripts*:

- *Public key.* The unlocking script must sign the transaction under this key.
- *Pay-to-public-key hash (P2PKH).* The unlocking script must provide a public key that hashes to the given value ("address"), and must then sign the transaction.
- *Multisignature.* An M-of-N ($N \leq 15$) multisignature scheme provides N public keys and requires M signatures in the unlocking script.
- *Pay-to-script hash (P2SH).* This script is the hash of a non-P2SH standard transaction. The unlocking script provides the full script hashing to this value and any necessary signatures. This script is typically used to shorten the length of multisignature transactions.
- *Data output (OP_RETURN).* The output cannot be redeemed but can be used to store up to 40 arbitrary bytes, such as human-readable messages.

For a transaction to be valid, it must contain all the required fields, all signatures must be correct, and the scripts must be standard. This is a task that miners undertake for a small fee. In addition, non-standard scripts using different sequences of `opcodes` can be included in blocks for higher fees. We discuss these in the context of a recent SHA1 collision later.

Mining and Consensus

To ensure that no coin is used more than once, every transaction is made public through a global, append-only ledger called the blockchain, consisting of *blocks* combining transactions in a Merkle tree. New blocks become a part of the blockchain through a process called mining: miners need to find a value (nonce) such that the hash of a block's header is less than a given target $h(hdr \,||\, nonce) < T$. The idea behind this PoW scheme is that the probability of creating the next block is proportional to the miner's computational power, and because miners receive transaction fees, they are incentivized to validate transactions and blocks, using the procedure shown in Figure 2.

Due to the probabilistic nature of mining, the presence of adversaries, and networking delays, miners may disagree on the current state of the blockchain. This is known as a *fork*. To deal with this issue, there

```
input : Bitcoin block
output: valid or invalid
/* Verify block header */
Verify Hash(block header) < target
Verify Merkle hash
Verify Hash(prev block) = prev_hash
/* Verify each transaction input in block */
foreach transaction input in the block do
    Check that referenced output transaction exists and hasn't
     already been spent
    Verify signatures
end
```

Figure 2. Procedure to verify a block's cryptographic primitives.

are hard-coded blocks included in the clients, known as *checkpoints*, starting from the first block, called the *genesis block*. In addition, honest (non-adversarial) miners work on the longest blockchain they become aware of, when other nodes announce new blocks and transactions.

These temporary forks enable double spending: an adversary can have different transactions in different branches of the fork using the same inputs but different outputs. However, because the probability of "deep" forks (forks where branches differ in the last N blocks) drops exponentially in N, receivers usually wait for multiple confirmation blocks.

Network

The last key component is the P2P network for distributed operation. Transactions and blocks are broadcast by nodes to their peers, and then relayed further to flood the network if they meet the relay policies (to prevent DoS attacks). Not every node is a miner or necessarily has access to the full chain: "lightweight" clients that use Simple Payment Verification (SPV) download only headers and the relevant transactions (with the corresponding Merkle trees).

Over time, the need for extensions or bug fixing motivates protocol changes. Because not all nodes upgrade at the same time, this may introduce forks. If the validation rules in the upgrade become stricter, then the protocol remains backward compatible, resulting in a *soft fork*. A *hard fork*, on the other hand, is not backward compatible and thus requires the entire network to upgrade, as old software would reject new transactions and blocks as invalid.

Broken Hashing Primitives

Here we look at the cryptographic hash functions in Bitcoin and analyze the effect of a break in one of the properties of first and second pre-image and collision resistance (defined below).

Hashing in Bitcoin

In the original Bitcoin paper,[2] the concrete primitives used are not specified: there were no "addresses" but just public keys, and the hash used for mining and the Merkle tree was just referred to as a hash function. The current Bitcoin implementation uses two hash functions.

Main hash. This hash function has an output of 256 bits and requires applying SHA256 twice: $H_M(x) = SHA256(SHA256(x))$. It is the hash used for mining (PoW): miners need to find a nonce such that the double SHA256 hash of a block header is less than a "target" hash. It is also used to hash transactions within a block into a Merkle tree, a structure that summarizes the transactions present within a block. Finally, it is the hash used for transactions signed with a user's private key.

Address hash. The second hash function is used as part of the P2PKH and the P2SH scripts. Its output is 160 bits, and it is concretely instantiated as $H_A(x) = RIPEMD160(SHA256(x))$.

Modeling Hash Breakage

Here we analyze how hashes break in terms of their building blocks.

Identifying hashing building blocks. A good cryptographic hash function $h(x)$ should offer three properties:

- *Pre-image resistance*—given y, it is hard to find x with $h(x) = y$;
- *Second pre-image resistance*—given x_1, it is hard to find $x_2 \neq x_1$ with $h(x_1) = h(x_2)$; and
- *Collision resistance*—it is hard to find distinct $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$,

where "hard" refers to computational infeasibility. This is because hash functions have a fixed-length output, so collisions always exist.

We consider attacks against H_A and H_M abstractly, so that our arguments can be extended for any future version that uses the same structure. Table 1 contains a summary of our results. We later relate the attacks we discover back to the concrete RIPEMD160 and SHA256 primitives.

Main Hash

Here we analyze the main hash H_M, which is used for mining, in Merkle trees, and with signatures. We discuss all three use cases separately.

Mining. We first investigate pre-image attacks against the block headers under two different attack scenarios,

before turning to collision and second pre-image attacks.

Attack 1: Pre-image against fixed Merkle root. We show that the probability that an adversary with access to a pre-image oracle can break mining is negligible. Miners search for block headers whose n-bit hash is below a target, which we assume starts with d zeros. This assumption introduces only up to 1 bit of extra work, as there is always a unique d with $T \leq 2^d < 2T$, for any target T.

If the adversary controls $b \leq n$ bits of the input, there are 2^b possible inputs to the hash function. These need to map to one of the 2^{n-d} values in the range $[0^n,0^d1^{n-d}]$ and will be uniformly distributed across 2^n values. This gives the expected number of b-bit pre-images as E [# pre-images] $= 2^b \cdot (2^{n-d})/(2^n) = 2^{b-d}$. The adversary can only query the pre-image oracle for specific target hashes. Because there are 2^{b-d} b-bit pre-images distributed across the 2^{n-d} values, the probability that a given hash in $[0^n,0^d1^{n-d}]$ has a b-bit pre-image is: P[correct pre-image] $= (2^{b-d})/(2^{n-d}) = 2^{b-n}$. This probability does not depend on d, as one might expect. This is because, by increasing d to reduce the number of valid hashes, the adversary also reduces the expected number of b-bit pre-images. Assuming the adversary is allowed 2^a queries to the oracle, the probability of breaking mining becomes P[success] $= 2^a \cdot 2^{b-n} = 2^{a+b-n}$.

To calculate b, we explore all fields in the block header. The version number (`nVersion`), the hashes of the previous block header (`hashPrevBlock`), and the hashes of the current Merkle root hash (`hashMerkleRoot`) are fixed. However, the adversary has partial control over the remaining fields in the header. For the timestamp field (`nTime`), the value can be within 7,200 seconds of the current median timestamp, giving the adversary approximately 13 bits of freedom. Moreover, the adversary has complete control over the 32 bits of the nonce (`nNonce`). The `nBits` field $0xAABBCCDD$ describes the target difficulty as $0xBBCCDD \cdot 256^{0xAA-3}$, with the protocol checking only that the produced number is at most the target value given by the consensus. At the time of writing, the target value is 0x1743eca9, granting the adversary approximately 27 bits of freedom.

Together the fields give $b = 72$. With $n = 256$, and allowing 2^{80} calls to the oracle, the probability of success is only $2^{80+72-256} = 2^{-104}$, which is negligible.

Attack 2: Pre-image against variable Merkle root. By varying the Merkle root, an adversary can break mining, although per the discussion of Attack 1, this cannot be achieved by simply reordering or excluding transactions. Instead, the adversary must work backward by querying the oracle for a target Merkle hash and repeatedly querying the oracle to reconstruct the entire Merkle tree. This would normally fail, as the transactions generated would not be valid due to incorrect signatures (although transaction malleability can allow the same signature to be valid for two different transactions[3]), but Bitcoin does not enforce a minimum number of transactions in a block. Hence, miners can mine blocks with just the coinbase transaction, which generates new coins and which has a variable-length input of up to 100 bytes that is controlled by miners. A malicious miner with access to the pre-image oracle can then:

- Pick an arbitrary target T and get a pre-image for $H_M(a||x||b) = T$, where the desired x is the `hashMerkleRoot` field and a,b are the remaining fields in a block header. Because the root is 256 bits, there is a pre-image with high probability, but if not, repeat with some other random target T'.
- Pick a length l for the script, and fix all other fields for the coinbase transaction. Solve $H_M(a'||y||b') = x$ where a',b' are the remaining fields for the coinbase transaction. Because the number of free bytes is up to 100, there is an l-bit pre-image y with high probability. The miner then generates a coinbase transaction using a',y,b' and combines it into a block using a,b. This block will have a hash of T as desired.

Table 1. Summary of the effects on Bitcoin for different types of hash breakage.

Breakage	Address hash (H_A)	Main hash (H_M)
Collision	Repudiate payment	Steal and destroy coins
Second pre-image	Repudiate payment	Double spend and steal coins
Pre-image	Uncover address	Complete failure of the blockchain

Collisions, second pre-images. Collisions and second pre-images are useful for mining only if the pre-images start with d zeros. Assuming the pre-images contain valid transactions and signatures, a miner can fork the chain, but this only occurs with negligible probability.

Merkle trees. Breaks of the main hash can also be used to attack Bitcoin through the Merkle trees used to store transactions in blocks.

Altering existing blocks. A similar argument as for mining (Attack 1) shows that an adversary cannot find a valid second pre-image of an entire block except with negligible probability. Pre-images do not give the

Table 2. Effects of a broken signature scheme.

Breakage	Effect
Selective forgery	Steal coins from public key
Integrity break	Claim payment not received
Repudiation	–

adversary new information, as they already accompany the hash value. Collisions are also not useful, as both values are attacker controlled and cannot alter existing blocks.

Attacking new blocks. For new blocks and transactions, an adversary with sufficient network control can use a collision or second pre-image to split the network, reject both blocks, or reverse transactions, thus enabling double spending. Pre-images are again not useful, as they always accompany the hashed value.

Main hash usage in signatures. In Bitcoin, signatures are over messages hashed with H_M. Therefore, a second pre-image attack or a collision on H_M can be used to destroy and possibly steal coins: an adversary can ask for a signature on an innocuous transaction (for example, pay 1 satoshi from address X to address Y) but transmit a malicious one instead (for example, pay 100 BTC from address X to address Z) because the adversary controls enough bytes to guarantee success with high probability. Note that for this attack to succeed, the adversary must still specify the same unspent transaction X belonging to the victim for the signature to be valid, but can alter the amount (bounded by the total number of bitcoins present in address X) and the destination.

Address Hash

The address hash is used in two contexts. First, in P2PKH scripts a Bitcoin address is essentially $y = H_A(p) = RIPEMD160\,(SHA256\,(p))$ where p is the public key (together with a checksum). Payments to addresses use only the hashed value y, but transactions from addresses require the full public key p and the signature on the transaction. The second use is in P2SH scripts. A P2SH is $y = H_A(s)$, where s is a standard script, typically a multisignature transaction. Payments to a P2SH script do not reveal the pre-image, but transactions spending the coins require it and the signatures of the corresponding parties. We discuss them jointly, since the only difference between a P2PKH and a P2SH in this context is the number of required signatures.

Pre-image. For previously spent outputs or for reused addresses, H_A is already accompanied by its pre-image. A pre-image thus can only reveal the public key(s) for unspent outputs. This has minimal privacy consequences because public keys are not tied to real identities, but it could enable an offline attack on the key. Assuming that the key was not chosen with bad randomness and there is no weakness in the signature scheme, the probability of success is still negligible.

Second pre-image. A second pre-image gives the adversary access to a different public key or script with the same hash. However, because the adversary does not control the corresponding private key, he or she cannot use this to change existing transactions or create new ones. This is because pre-images (whether a key or a script) are revealed and verified only when spent in transactions.

Collision. Collisions are similar, although in this case, both public keys are under the adversary's control and again the adversary does not have access to the private keys. In both scenarios, there is a question of non-repudiation external to the protocol itself: by presenting a second pre-image of a key used to sign a transaction, a user/adversary can claim that his or her coins were stolen.

Broken Signature Primitives

Here we describe the use of digital signatures in Bitcoin and analyze how a break in their unforgeability, integrity, or non-repudiation impacts Bitcoin. Table 2 summarizes our results.

Digital Signatures in Bitcoin

Bitcoin's digital signature scheme is ECDSA with the `secp256k1` parameters, and is used to sign the main hash H_M of transactions.

Modeling Signature Breakage Variants

The security of digital signature schemes is usually discussed in terms of three properties, which we define as follows:

- *Unforgeability*—no one can sign a message m that validates against a public key p without access to the secret key s;
- *Integrity*—a valid signature $\{m\}_s$ does not validate against any $m' \neq m$; and
- *Non-repudiation*—a valid signature $\{m\}_s$ does not validate against any public key $p' \neq p$,

where there is an implicit "except with negligible probability," due to hashing.

These properties are linked, and a breakage in one usually implies a breakage in the others. In addition, they are often discussed in a much more abstract way: non-repudiation refers to the property that the signature proves to all parties the origin of the signature, but in this case, we introduce it in a way that is more akin to Duplicate Signature Key Selection (DSKS) attacks.[4]

Broken Signature Scheme Effects

We now analyze a break in each of these properties separately, starting with the last two, as neither of them can lead to an attack on their own.

Integrity. For a break in the integrity of the signature scheme to be useful in Bitcoin, a signature of $H_M(m)$ must also be valid for $H_M(m')$. This involves H_M in a non-trivial way, so we discuss this further later, but note that transaction malleability can cause the issuer of a transaction to think that his or her payment was not confirmed.[3]

Non-repudiation. For non-repudiation, we note that for transactions, even if a signature verifies under a different key, the address hashes of the two public keys must match. A break thus involves H_A, so we also discuss this case further below.

Unforgeability. When it comes to unforgeability, we can distinguish between various types of breaks:[5] *total break* to recover the private key, *universal forgery* to forge signatures for all messages, *selective forgery* to forge a signature on a message of the adversary's choice, and *existential forgery* to produce a valid signature that is not already known to the adversary.

Because the message to be signed must be the hash of a valid transaction, an existential forgery is not sufficient because the probability that it corresponds to a valid message is negligible. Selective forgery, on the other hand, can be used to drain a victim's wallet. From this perspective, selective forgery and a total break have the same effect. However, as we discuss later, the type of breakage influences how to upgrade to a new system. It is worth noting that an adversary does not necessarily have access to a user's public key, because addresses that have not been reused are protected by the address hash H_A.

Multi-Breakage

Here we analyze how combinations of breakages in different primitives can impact Bitcoin. Because H_A and H_M are not used together, we consider only a break in the signature algorithm in combination with a break in one of the two hashes. The results are summarized in Table 3.

Table 3. Multi-breakage effects: combining broken hashes and signatures.

	Signature property		
Hash property	**Selective forgery**	**Integrity break**	**Repudiation**
Address hash (H_A)			
Collision	Repudiate transaction	-	Change existing payment*
Second pre-image	Steal all coins	-	Change existing payment
Pre-image	Steal all coins	-	-
Main hash (H_M)			
Collision	Steal coins	Steal coins*	-
Second pre-image	Steal coins	Double spend*	-
Pre-image	-	-	-

** Achieving this requires a modification of definitions. See text for details.*

Address Hash and Signature Scheme

A combined break of the address hash and the signature scheme allows an adversary to steal coins, and repudiate or replace transactions.

Signature forgery. A combination of a selective forgery with a first or second pre-image break of the address hash can be used to steal all coins that are unspent. Generating two public keys p,p' with $H_A(p) = H_A(p')$ (collision) whose signatures the adversary can forge does not have a direct impact, because the adversary controls both addresses. However, it appears as if two different users are attempting to use the same coin, thus raising a question of repudiation, which we discuss later.

Signature integrity. As the messages signed for transactions do not involve H_A, this combination does not increase the adversary's power.

Signature repudiation. A pre-image attack on H_A is not useful as the public key is already known. For a second pre-image, assume that given a message m (the hash of a transaction) and a public key p, an oracle returns p' such that $H_A(p) = H_A(p')$ and the signature of m under p also validates against p'. Because the same signature validates for both keys, an adversary can replace p by p' in the unlocking script. Although this does not give the

Table 4. Effects of concrete primitive breakage on the current version of Bitcoin.

Breakage	Effect
SHA256	
Collision	Steal and destroy coins
Second pre-image	Double spend and steal coins
Pre-image	Complete failure
RIPEMD160	
Any of the above	Repudiate payments
ECDSA	
Selective forgery	Steal coins
Integrity break	Claim payment not received
Repudiation	-

adversary immediate monetary gain, a transaction in the blockchain has been partially replaced.

For collisions, assume that given a message m, an oracle returns two public keys p,p' such that $H_A(p) = H_A(p')$ and the signature of m under p validates under p'. If the adversary does not have access to the private keys, he or she cannot sign the transaction. Otherwise, the effect is identical to the second pre-image case, where the adversary can replace part of a transaction in the blockchain.

Main Hash and Signature Scheme

A combined break of the main hash and the signature scheme allows an adversary to steal or double spend coins.

Signature forgery. As explained earlier, none of the potential attacks using the hash H_M required a break in the signature scheme. The partial exceptions were mining under a pre-image break and transactions with second pre-image or collision breaks. We discuss each possibility below.

For mining, a pre-image attack is useful when working backward from a fixed target to get a pre-image for the Merkle root and turn it into a tree of transactions. The problem we identified earlier was that there is only negligible probability that the transactions refer to valid, unspent outputs, so a forgery does not solve this issue. Finally, for transactions, collisions and second pre-images on their own can be used to destroy or steal coins. If adversaries can also forge signatures, they are guaranteed to be able to steal coins no matter what address they went to, as long as it is not protected by the address hash.

Signature integrity. A collision or a second pre-image attack trivially breaks the integrity of the scheme, as messages are always hashed, and reduces to the case discussed earlier about main hash usage in signatures. We thus modify the definitions slightly to consider a joint break in the two algorithms.

A *collision integrity* oracle given a public key p produces m,m' such that the signature of $H_M(m)$ is also valid for $H_M(m')$. The adversary can ask for a signature on an innocent transaction but transmit the malicious one with the still valid signature. Unlike in the regular collision case, the two hashes $H_M(m)$ and $H_M(m')$ are different. Hence, the adversary cannot just replace the transaction in the block, but he or she can opt never to transmit the innocent one instead.

A *second pre-image integrity* oracle given a public key p and a message m produces m' such that the signature of $H_M(m)$ is also valid for $H_M(m')$. This case also resembles the break on just H_M, but again, because the hashes are not equal, the adversary cannot simply replace an existing transaction, unless it has not yet been confirmed in a block. This can split the network and destroy coins.

Signature repudiation. The non-repudiation property of the signature scheme necessarily involves a break of H_A, as explained earlier. This combination therefore does not increase the adversary's power.

Current Bitcoin Implementation

Here we revisit the current Bitcoin implementation, in the context of its choice of primitives, non-standard scripts, and contingency plans, using observations from the previous sections.

Current Cryptographic Primitives

In the current implementation of Bitcoin, $H_A(x) = RIPEMD160(SHA256(x))$, and $H_M(x) = SHA256(SHA256(x))$. Because there are no critical breaks for H_A, a break in RIPEMD160 is not cause for concern. Moreover, because H_M uses only SHA256, an attack against SHA256 is equivalent to an attack against H_M. We can thus summarize the effect of concrete primitive breakage, shown in Table 4.

Non-Standard Scripts

Again, the Bitcoin scripting language extends beyond the five standard types of transactions. In part due to the higher fees associated with them, non-standard transactions represent less than 0.02 percent of all

bitcoins in circulation (https://p2sh.info/dashboard/db/non-standard-outputs-statistics) but are more versatile and can include opcodes for calculating SHA1, SHA256, and RIPEMD160 hashes. By using these opcodes, one can create "challenges" involving the primitives, so that, for instance, a user needs to create a collision to redeem a certain coin.

One set of such challenges was created by an early Bitcoin developer asking for collisions on individual and combined primitives (https://bitcointalk.org/index.php?topic=293382.0). The challenge for a SHA1 collision was claimed using the collision found by Stevens and colleagues[6] for a bounty of 2.48 BTC, the equivalent of approximately $2,800 at the time. Although the USD/BTC exchange rate fell temporarily as a result of this news, it quickly recovered, because SHA1 is not an integral part of the Bitcoin protocol. However, this collision highlights the need to anticipate the breakage of primitives, because non-standard transactions allow collision and pre-image attacks to be used even when the core protocol is not yet broken. However, as we explain, the existing contingency plans are not sufficient.

Existing Contingency Plans

A break of the primitives has interested the community from the early days of Bitcoin. Informal recommendations by Satoshi in forums evolved into a wiki page that describes contingency plans for "catastrophic failure[s]."[1] Such a failure for primitives is defined in terms of an adversary that can defeat the algorithm with "a few days of work,"[1] and the focus is on notifying users and protecting the `OP_CHECKSIG` operation to prevent people from stealing coins.

Concretely, for a "severe, 0-day failure of SHA-256,"[1] the plans propose switching to a new hashing algorithm H' and hard-coding known public keys with unspent outputs as well as the Merkle root of the blockchain under H'. For a broken signature scheme, if the attacker cannot recover the private key and there is a drop-in replacement using the same key pair, the plan is to simply switch over to the new algorithm. Otherwise, the new version of Bitcoin "should automatically send old transactions somewhere else using the new algorithm."[1]

Potential Migration Pitfalls

The contingency plans suggest that "code for all of this should be prepared,"[1] but no such mechanism currently exists. Moreover, no plans are in place for a break in RIPEMD160. Because sudden breaks are unlikely, neither is cause for immediate concern, but should be included in future plans.

Broken SHA256. By our analysis, it is clear that new transactions should not use a broken hash. However, existing historical transactions and blocks cannot be altered, except in a majority mining attack. Thus, hard-coding public keys and rehashing the entire blockchain are more prudent than necessary. It should be noted that a sudden break necessitates a hard fork for Bitcoin.

Broken ECDSA. For a broken ECDSA, a transition is indeed easy if there is a drop-in replacement and the private key is safe. Otherwise, a gradual transition scheme is necessary as users will need to manually switch over to a new key pair.

Recommendations

Here we make recommendations to more properly anticipate primitive breakage. Recognizing that there are financial considerations in addition to the technical ones, we do not propose a full upgrade mechanism but merely make suggestions to the Bitcoin developers and community.

First of all, our analysis reinforces the idea that users should not reuse addresses, not just for privacy reasons but also because this protects against some types of primitive breakage. For instance, if the signature scheme is broken, addresses are still protected by the hash.

The plans for a sudden breakage should address when to freeze the blockchain and whether to roll back transactions in the case of a sudden break. Moreover, the centralized approach of hard-coding well-known keys is perhaps not entirely in line with Bitcoin's decentralized philosophy and can lead to lost coins. If keys are to be hard-coded, there is a tradeoff between complexity and risking making coins unspendable: developers must decide whether the migration would occur at once or whether new key pairs should be distributed periodically. An alternative and perhaps better approach would be to use zero-knowledge proofs to tie the old address still protected by its hash to the new public key.

Given that sudden breaks are unlikely, there is a need for a separate plan for weakened primitives. Based on our analysis, we recommend the following:

- Introduce a minimum number of transactions per block to increase the difficulty of performing the pre-image attack against the mining header target (PoW) using the coinbase transaction.
- To migrate from old addresses, whether due to a weakened hash or signature scheme, introduce new address types using stronger hashing and signature schemes. This can be achieved with a soft fork by making transactions appear to old clients as "pay-to-anybody," akin to how P2SH was introduced.
- Instead of using nested hashes for H_A, H_M, combine primitives in a way that increases defense in depth (see the Related Work sidebar).

- Given that H_M has multiple use cases, consider whether each of its functions should have a different instantiation, whether through distinct primitives, by prepending different values, or by using an HMAC with different keys.
- Consider a hard fork in response to a weakened H_M, with redesigned headers and transactions and without any use of the old primitives.

A soft fork is insufficient for properly upgrading a weakened hash function $H_M = H_1$ to the stronger H_2, because H_M forms the core of the PoW scheme. Specifically, because any changes must be backward compatible, the old validation rules must still apply, so for every new block, $H_1(hdr) < T$, where the target T is still calculated by the same algorithm. New blocks would also need to satisfy some additional constraint $H_2(hdr') < T'$, where the target T' is calculated independently and hdr' is the block header, possibly excluding some fields. As a result, new clients would have to solve two PoW computational puzzles. Although every instance of H_1 (transaction, Merkle root, and so on) could be accompanied by an instance of H_2, blocks and transactions are fundamentally identified by their H_1 hash, which an attacker could exploit. There are also questions of incentives and whether new iterations of Bitcoin would still use a PoW scheme, but this is left as future work.

Other Cryptocurrencies

Bitcoin is the largest cryptocurrency by market capitalization and adoption, but both derivatives ("altcoins") and completely different designs have spread. Although discussing them all is out of scope, we identify a few common threads among those that use PoW schemes.

The first class of altcoins consists of those that are forks of Bitcoin with additional types of transactions supported. For these currencies, our analysis applies verbatim but needs to be extended to these new types of operations. As an example, Namecoin introduces transactions for registering and updating `.bit` domains. The `name_new d/<name>` command creates a transaction whose outputs include $H_A(r||\text{name})$, where r is a nonce. This acts as a pre-order for the domain `name.bit`, which is then updated by its owner after the transaction has been in 12 confirmed blocks. This pre-order commitment phase is necessary to ensure no one can steal the name of the domain, which is fully registered in a subsequent `name_firstupdate` command that reveals the name, the nonce, and the DNS configuration values, signing the transaction with the user's key.

Any updates for a domain require signatures, so any novel exploit would involve the `name_new` and `name_firstupdate` commands. Unlike Bitcoin, however, a pre-image attack on H_A allows recovering a random nonce and domain name. Namecoin rules do not seem to preclude *name_new* transactions with the same hash, but if the pre-image recovers the original (*r*, *name*) pair, an attacker can pre-register the domain by publishing his or her own `name_new` transaction before the original user's transaction is confirmed. In practice, however, this attack would require pre-image oracles for both hash functions used in H_A (SHA256, RIPEMD160), implying that there is a pre-image oracle for H_M as well, making the entire currency insecure.

The second set of cryptocurrencies consists of those that separate the roles of H_M and replace it by two different hash functions, say H_I for integrity checking and H_P for the PoW. If H_I breaks, one can still steal coins by creating collisions in transactions, which are hashed before they are signed. However, because H_I is not used for PoW, an attack on H_I cannot directly be used to exploit mining. More importantly, as explained earlier, even a full pre-image oracle on H_P is not sufficient for a breakdown of the currency. This is because the probability of finding a valid pre-image for a fixed target is negligible, as the adversary does not control sufficiently many header bits. Instead, as also identified earlier, an adversary needs to be able to alter the Merkle root at will, which is protected by the different H_I, requiring both to be exploited for a successful attack. Example currencies employing this scheme include Litecoin and Dogecoin, where $H_I = \text{SHA256}(\text{SHA256})$, and H_P is the `scrypt` key derivation function. Ethereum is similar, with H_I using Keccak-256 and H_P using Ethash, although it should be noted that due to the different structure of the blockchain, our results are not directly applicable, even though our methodology is.

Primecoin has a different PoW mechanism. It is a fork of Bitcoin using the same structures (and only the single H_M), but its PoW requires finding long chains of prime numbers p_i that satisfy $p_{i+1} = 2p_i \pm 1$, with $p_0 = k{\cdot}H_M(\text{header}) \pm 1$ for some k (technically, Cunningham chains of first or second kind, or bi-twin chains). An adversary with access to a pre-image oracle can attack this scheme by reusing existing chains: if (p_i) is a valid chain for header h and the difficulty has not changed, then (p_i) is also a valid chain for h/q for any factor q of h, and for $h{\cdot}r$, where r is a small factor of k (to ensure that $h{\cdot}r$ remains a valid hash). Note that once more, this exploit depends on using the pre-image oracle twice: once for the overall header and once for the coinbase transaction, showing that it is necessary to increase the minimum number of transactions per block.

Finally, there are currencies that use a hybrid PoW and proof-of-stake or proof-of-burn scheme. For instance, Peercoin is a fork of Bitcoin that uses the same PoW and structure as Bitcoin, but also allows "minting" coins based on their age for a proof-of-stake approach. As a result, the attacks identified in this article still apply,

Related Work

Identifying how hash collisions break the security of protocols such as TLS, IPSec, and SSH was recently investigated in "Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH."[7] More recently, researchers identified vulnerabilities with the cryptographic function used in the IOTA cryptocurrency, which allowed them to create syntactically valid transactions with the same hash, but which pay out different amounts.[8] In terms of the primitives used in Bitcoin, attacks against RIPEMD160 pre-images[9] and collisions[10] as well as SHA256 collisions[11] and pre-images[12] work only for a reduced number of rounds and incrementally improve on brute-force solutions. Certain Elliptic Curve Digital Signature Algorithm (ECDSA) parameters can lead to Duplicate Signature Key Selection, where an adversary can create a different key P' that validates against a correct signature under a key P.[4] Such research indicates that Bitcoin primitives are indeed under attack, highlighting the need for anticipating their breakage.

More generally, for combining hashes effectively, "Multicollisions in Iterated Hash Functions: Application to Cascaded Constructions"[13] shows that simultaneous collisions for multiple hash functions are not much harder to find than individual ones. "On the Strength of the Concatenated Hash Combiner When All the Hash Functions Are Weak"[14] shows that even when the underlying compression functions behave randomly but collisions are easy to generate, finding collisions in the concatenated hash $h_1(x)||h_2(x)$ and the XOR hash $h_1(x) \oplus h_2(x)$ requires $2^n/2$ queries. However, when the hash functions use the Merkle-Damgård (MD) construction, there is a generic pre-image attack against the XOR hash with complexity $\tilde{O}(2^{5n/6})$.[15] MD hash functions also behave poorly against pre-image attacks, allowing one to find second pre-images of length 2^{60} for RIPEMD160 in $2^{106} \ll 2^{160}$ time.[16] If an adversary can further find many collisions on an MD construction, he or she can also find pre-images that start with a given prefix (Chosen Target Forced Prefix).[17]

but the attack surface increases to include the new transactions and needs to be analyzed separately, as we did with Namecoin above.

We presented the first systematic analysis of the effect of broken primitives on Bitcoin. Our analysis reveals that some breakages cause serious problems, whereas others are inconsequential. The main vectors of attack involve collisions on the double SHA256 hash or attacking the signature scheme, which directly enables coin stealing. In contrast, a break of the hash used in addresses has minimal impact, because addresses do not meaningfully protect the privacy of a user. Our analysis has also uncovered more subtle attacks. For example, the existence of another public key with the same hash as an address in the blockchain enables parties to claim that they did not make a payment. Such attacks show that an attack on a cryptographic primitive can have social rather than technical implications. We leave the economic impact of such attacks as future work. Because our analysis abstracts away from the concrete primitives, our general results extend to future versions that use a similar structure as well as altcoins and other blockchain-based schemes.

We uncovered a lack of defense in depth in Bitcoin. In most cases, the failure of a single property in one cryptographic primitive is as bad as multiple failures in several primitives at once. For future versions of Bitcoin, we recommend including various redundancies, such as properly combined hash functions, and requiring a minimum number of transactions per block. Bitcoin's migration plans are currently underspecified and offer at best an incomplete solution if primitives get broken. We offer some initial guidelines for making the cryptocurrency more robust not only for a sudden break but also in response to weakened primitives. However, future discussions should directly involve the Bitcoin developers and community to propose plans that would be in line with their expectations. ■

Acknowledgments

This article extends a paper presented at ESORICS 2016.[18]

References

1. "Contingency Plans," Bitcoin Wiki, 2015; https://en.bitcoin.it/wiki/Contingency_plans.
2. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008; https://bitcoin.org/bitcoin.pdf.
3. C. Decker and R. Wattenhofer, "Bitcoin Transaction Malleability and MtGox," *European Symposium on Research in Computer Security* (ESORICS), 2014.
4. S. Blake-Wilson and A. Menezes, "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol,"

International Workshop on Practice and Theory in Public Key Cryptography (PKC), 1999.
5. S. Goldwasser, S. Micali, and R.L. Rivest, "A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks," *SIAM Journal on Computing* (SICOMP), 1988.
6. M. Stevens et al., "The First Collision for Full SHA-1," *Annual International Cryptology Conference* (CRYPTO), 2017.
7. K. Bhargavan and G. Leurent, "Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH," *Network and Distributed System Security Symposium* (NDSS), 2016.
8. E. Heilman et al., "IOTA Vulnerability Report: Cryptanalysis of the Curl Hash Function Enabling Practical Signature Forgery Attacks on the IOTA Cryptocurrency," 7 Sept. 2017; https://github.com/mit-dci/tangled-curl/blob/master/vuln-iota.md.
9. C. Ohtahara, Y. Sasaki, and T. Shimoyama, "Preimage Attacks on Step-Reduced RIPEMD-128 and RIPEMD-160," *International Conference on Information Security and Cryptology* (Inscrypt), 2010.
10. F. Mendel et al., "Improved Cryptanalysis of Reduced RIPEMD-160," *International Conference on the Theory and Application of Cryptology and Information Security* (ASIACRYPT), 2013.
11. F. Mendel, T. Nad, and M. Schläffer, "Improving Local Collisions: New Attacks on Reduced SHA-256," *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (EUROCRYPT), 2013.
12. D. Khovratovich, C. Rechberger, and A. Savelieva, "Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family," *International Workshop on Fast Software Encryption* (FSE), 2012.
13. A. Joux, "Multicollisions in Iterated Hash Functions: Application to Cascaded Constructions," *Annual International Cryptology Conference* (CRYPTO), 2004.
14. J.J. Hoch and A. Shamir, "On the Strength of the Concatenated Hash Combiner When All the Hash Functions Are Weak," *International Colloquium on Automata, Languages and Programming* (ICALP), 2008.
15. G. Leurent and L. Wang, "The Sum Can Be Weaker Than Each Part," *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (EUROCRYPT), 2015.
16. J. Kelsey and B. Schneier, "Second Preimages on n-bit Hash Functions for Much Less Than 2^n Work," *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (EUROCRYPT), 2005.
17. J. Kelsey and T. Kohno, "Herding Hash Functions and the Nostradamus Attack," *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (EUROCRYPT), 2006.
18. I. Giechaskiel, C. Cremers, and K.B. Rasmussen, "On Bitcoin Security in the Presence of Broken Cryptographic Primitives," *European Symposium on Research in Computer Security* (ESORICS), 2016.

Ilias Giechaskiel is a DPhil (PhD) student in the Department of Computer Science at the University of Oxford, focusing on hardware security. He holds an MPhil in advanced computer science from the University of Cambridge and a BA in mathematics from Princeton University. He has also interned at Bloomberg, Microsoft, Dropbox, Microsoft Research, and Jump Trading, often taking on security-oriented projects. Contact at ilias.giechaskiel@cs.ox.ac.uk.

Cas Cremers is a full professor in the Department of Computer Science at the University of Oxford. His research focuses on information security, in particular the formal analysis of security protocols. This work ranges from developing mathematical foundations for protocol analysis to the development of analysis tools, notably the Scyther tool and the Tamarin prover. He has also been involved in protocol standardization, including the improvement of the ISO/IEC 9798 & 11770 and IETF TLS 1.3 standards, and applied cryptography, leading to the development of new security requirements and protocols. Contact at cas.cremers@cs.ox.ac.uk.

Kasper B. Rasmussen is an associate professor in the Computer Science Department at the University of Oxford. He joined the department in 2013 and in 2015 was awarded a University Research Fellowship from the Royal Society in London. Prior to being at Oxford, Rasmussen spent two years as a post-doc at University of California, Irvine. He did his PhD with prof. Srdjan Capkun at the Department of Computer Science at ETH Zurich (Switzerland), where he worked on security issues relating to secure time synchronization and secure localization with a particular focus on distance bounding. His thesis won the "ETH Medal" for an outstanding dissertation from the Swiss Federal Institute of Technology, and he was awarded the Swiss National Science Foundation (SNSF) Fellowship for prospective researchers. Contact at kasper.rasmussen@cs.ox.ac.uk.

PQChain: Strategic Design Decisions for Distributed Ledger Technologies against Future Threats

Rachid El Bansarkhani, Matthias Geihs, and Johannes Buchmann | Technische Universität Darmstadt

In this work, we make strategic design decisions for a secure instantiation of the blockchain protocol, taking into account the presence of large-scale quantum computers and potential future attacks against the underlying hash functions. Furthermore, we highlight that considering the blockchain protocol as a cryptographic primitive can help improve other cryptographic primitives.

Current research in distributed ledger technologies deals with a variety of technical challenges; however, its vulnerability to quantum computer attacks has not been investigated so far, even though many of the applied digital signature schemes to authenticate transactions are based on classical assumptions such as the discrete log problem, which can be broken by Peter Shor's algorithm. In this work, we analyze distributed ledger technologies with regard to its intended application area. Subsequently, we identify key design features to be addressed for an appropriate and secure blockchain instantiation.

Background

Blockchain technology is celebrated for its huge economic impact and its wide variety of applications penetrating almost all industrial sectors. Previously, consensus was reached via a trusted centralized authority such as a bank, a notary, a certificate authority, or any other trusted service. But since the first publication on Bitcoin under the pseudonym Satoshi Nakamoto in 2008,[1] many protocols and applications were based on Bitcoin's main building block, namely the blockchain protocol. It puts unknown network participants into the powerful position of achieving consensus solely based on cryptography and the fact that the majority of powerful miners are honest. There is no need for a trusted centralized service anymore, so many procedures in practice can be carried out much faster, leading to cost savings and more efficient business processes.

Motivation

It may take several years before protocols are in wide use. In the same time frame, the first large-scale quantum computers are expected to enter the market, threatening the security of asymmetric cryptography. In 1994, Peter Shor proposed[2] a quantum algorithm that could break all applied public-key cryptosystems in probabilistic polynomial time. Hence, security mechanisms based on the Rivest-Shamir-Adleman (RSA) cryptosystem and elliptic curve cryptography (ECC) must be avoided to protect confidentiality and

1540-7993/18/$33.00 © 2018 IEEE Copublished by the IEEE Computer and Reliability Societies

integrity. However, one also has to take care of the security of many symmetric key ciphers and hash functions, whose security is significantly reduced due to Grover's search quantum algorithm. Although the latter schemes require only an adjustment of parameters, the question of how to replace public-key cryptosystems remains open. Furthermore, what if a hash function becomes insecure due to an advanced attack many years after introducing blockchains into practice? This technology is applied in areas that are supposed to be sensitive to attacks, such as the healthcare and finance sectors, which may cause extensive damage with a tremendous negative impact on our society.

Applications of the Blockchain Protocol

Starting with the Bitcoin protocol, many new protocols have been invented. The main objectives can be classified into two subgoals. First, the blockchain system underlying the Bitcoin protocol is very inefficient in terms of power consumption and the overall generation time of new blocks, due to the associated proof of work (PoW) that is utilized to make the overall protocol work. Thus, researchers and practitioners have worked on improving the underlying protocol, leading to a series of new ideas such as proof of stake (PoS) and proof of elapsed time, just to name a few. These aim to reduce the power consumption and generation time of new blocks. Combinations of those principles are found in, for instance, Bitcoin NG,[3] a follow-up of Bitcoin. Second, currently applied systems are screened with regard to the applicability of the blockchain protocol to discover new opportunities or to replace existing centralized systems by protocols based on distributed consensus. This may allow improvement in the companies' performance or profitability. As a result, we find decentralized proof of existence of documents (https://proofofexistence.com), Ethereum (https://www.ethereum.org) realizing powerful smart contracts, Storj.io (https://storj.io) for distributed cloud storage, Namecoin (https://namecoin.org) for distributed DNS services, CertCoin[4] for decentralized public-key infrastructure (PKI) services, and Blockstack,[5] which combines several decentralized Internet services. There are many further application areas such as the Internet of Things, pharmaceutics, and finance sectors, aiming to introduce the blockchain protocol as a key technology. We refer the reader to the recent report of the World Economic Forum (http://www3.weforum.org/docs), illustrating various important future use cases exploiting smart contracts.

Preliminaries

Preliminaries include hash functions and blockchain protocols.

Hash Functions

A cryptographic hash function $H{:}\{0,1\}^* \rightarrow \{0,1\}^n$ is an efficiently computable function that maps strings of arbitrary length to strings of fixed length. Depending on the given application scenario, a certain set of properties might be required:[6]

- *One-wayness*: It is infeasible to find a pre-image m of any given hash value.
- *Collision resistance*: It is infeasible to find two distinct messages mapping to the same hash value.
- *Pseudorandomness*: It is infeasible to distinguish H from a uniform random function f.

We refer readers to "Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance"[6] for an overview of further properties of hash functions.

Blockchain Protocols

A blockchain protocol allows parties in a distributed network to achieve consensus on events that occurred in the network. It establishes a *public*, *immutable*, and *ordered* ledger of transactions in the network.[7] For example, blockchains are used to prevent double spending in digital currencies by ensuring that all parties have a consistent view of which coins have already been spent.

Achieving consensus in a distributed network with untrusted nodes in general is considered a very difficult problem in computer science.[8] Blockchains offer a solution to this problem by giving incentives to honest parties while punishing misbehaving parties. However, a blockchain protocol can function properly only as long as at least a certain majority of the network is not malicious.

Building Blocks

Building blocks include hash-based digital signatures, hash combiners, and blockchain PKI.

Hash-Based Digital Signatures

Hash-based digital signature schemes represent promising postquantum alternatives that are based on a collision-resistant hash function H with the following ingredients. For generation and verification of signatures, a Merkle hash tree (Figure 1) is used.[9]

Each leaf of that tree corresponds to a one-time signature (OTS) key pair, and the maximum number of signatures that can be generated using the scheme is limited by the number of leaves $N = 2^h$ in a tree with h layers. In the following, we describe the algorithms of the fundamental hash-based signature scheme.[9] A more advanced scheme with additional

security properties and improved efficiency is described, for example, in "XMSS—A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions."[10]

KeyGen. For key generation, N OTS key pairs (X_i, Y_i) are generated, and for each pair (X_i, Y_i), a hash value $h_i = H(Y_i)$ is computed. Then, a Merkle hash tree is computed from leaves $h_1, \ldots, h_N$. The private key is the set of key pairs $\{(X_i, Y_i)\}_{i \in [n]}$, and the public key is the root hash value r of the generated Merkle tree.

Sign. On input of a message m, the signing algorithm chooses an OTS key pair (X_i, Y_i) that has not been used for signing yet. It then uses the one-time signature scheme with the secret key X_i to sign m and obtain an OTS s. It outputs the signature s, the one-time public key Y_i, and the authentication path a, consisting of all the sibling nodes, from leaf i to the root of the Merkle tree.[9,10] We remark that the signature generation algorithm needs to maintain the state i for keeping track of which key pairs have already been used, and which can still be used for new signatures. This state is public information.

Verify. On input of a message m, an OTS s, a one-time public key Y, an authentication path a, and a public key root hash value r, the verification algorithm checks that s is a valid signature for m under public key Y and that a is a valid authentication path from $H(Y)$ to the root r.

Hash Combiners

A hash combiner Comb^{H_1,H_2} combines two hash functions H_1 and H_2 such that a given hash function property (for example, collision resistance) is preserved as long as at least one of the hash functions H_1 or H_2 has the property. We say a hash combiner $\mathrm{Comb}^{H_1,H_2}: \{0,1\} \rightarrow \{0,1\}^l$ is (strongly) robust for properties P, if for any property $p \in P$, if H_1 or H_2 has property p, then Comb^{H_1,H_2} has property p.

A hash combiner Comb^{H_1,H_2} is called weakly robust for properties P if properties are only guaranteed to be preserved together. That is, if H_1 or H_2 has every property in P, then Comb^{H_1,H_2} has properties P.

The concatenation combiner $\mathrm{Comb}_{\|}^{H_1,H_2}(M) = H_1(M) \| H_2(M)$ is a simple example of a hash combiner that is robust for collision resistance. This combiner, however, is not robust for pseudorandomness[11] and lacks efficiency in terms of output length.

Fischlin and colleagues proposed a variety of efficient hash combiners.[11] Their most efficient combiner Comb_{4P} has output length $2l$ and is robust for the properties collision resistance, pseudorandomness, target collision resistance, and message authentication. They also present a combiner $\mathrm{Comb}_{4P\&OW}$ that additionally preserves one-wayness at the cost of additionally computing a pairwise independent permutation.

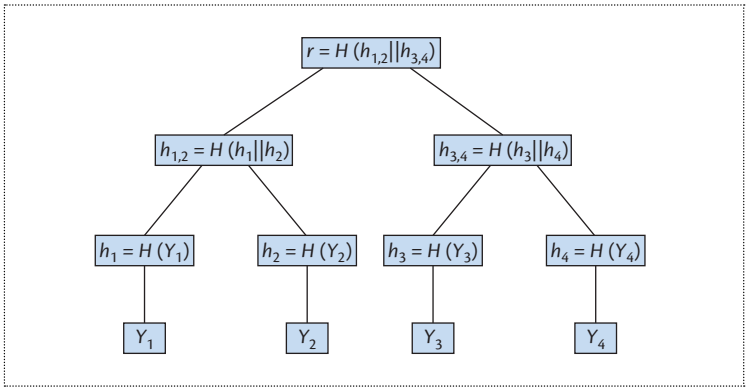


Figure 1. A Merkle tree for $N = 4$ leaves. The authentication path for leaf Y_1 is $(h_2, h_{3,4})$.

Blockchain PKI (Certcoin)

Certcoin is a PKI built on the basis of a blockchain protocol.[4] The idea is to publish public-key-registration, update, or revocation information in the blockchain as opposed to certificate authorities carrying out these tasks in traditional public-key infrastructures. The blockchain PKI provides the following core functionalities.

Register. A client registers his or her identity with his or her public key in the blockchain PKI. The identity and the public key are published in the blockchain if there is no key associated with that identity yet.

Update. A client updates the public key registered with his or her identity. The client signs the new public key with the old secret key and publishes the new public key together with the signature in the blockchain. The miners accept the public key update once the validity of the signature is verified by use of the old public key.

Revoke. A client revokes a public key registered with his or her identity. The client signs a revocation statement with its secret key and publishes the signed statement in the blockchain. The miners accept the revocation if the signature on the statement is valid with regard to the corresponding public key.

LookUp. A client retrieves the latest public key of a given identity.

Design Strategy for PQChain

Here, we give a detailed description of our design decisions for a new blockchain protocol, which requires

ensuring integrity, unforgeability, and authenticity for a long time period. A strategy has to take into account potential threats in the future and associated countermeasures. The Bitcoin protocol, for instance, has now been operating for almost nine years, which is not suitable for many real-world applications as the applied security measures do not fully comply with the ones required. There are several questions to be scrutinized prior to building a customized blockchain protocol. We classify them into the following categories. We note that employing a certain measure may impact all categories.

- *Purpose*: What is the goal of the blockchain protocol? Which application scenario has to be covered? Do legal aspects play a role?
- *Life cycle*: How long is the blockchain protocol aimed to guarantee certain (security) features—5 years, 50 years, or longer?
- *Reliability and efficiency*: What levels of efficiency and reliability are required for the target scenario? The underlying crypto tools, the P2P network, storage requirements, and other technical details must be considered together.
- *Risk of potential threats*: What are the potential threats that blockchain protocols may face during the time of operation? This includes Moore's law (if applicable), quantum computers, algorithmic improvements, and cryptanalysis of hash functions or other employed crypto primitives.
- *Reactive blockchain*: Can the blockchain protocol respond quickly to potential future threats?

> **A strategy has to take into account potential threats in the future and associated countermeasures.**

Based on this classification, our goal is to design a robust blockchain protocol that implements several preemptive measures to enhance the security of the overall system. The resulting protocol is hence applicable to a wider range of applications. However, we will not address each aspect, which may also depend on a specific application scenario.

Postquantum primitives. Due to recent developments and the tremendous amount of research effort in quantum computing, there is confidence that the first large-scale quantum computers will be available within the next 10 years. This coincides with the time frame expected for the deployment of blockchain-based technologies in real world scenarios. As a result, the blockchain protocol has to be equipped with security features that are resistant to quantum attacks. Despite the fact that the blockchain protocol inherently relies on hash functions, which are believed to be quantum immune, Grover's search algorithm allows mounting brute-force attacks in time $2^{n/2}$, where n is the output length. Hence, it is recommended to double the output length to preserve the same level of security.

Digital signatures. Quantum computer attacks not only decrease the security of the underlying hash function, but also break all currently applied public-key cryptosystems. For instance, the Bitcoin protocol applies Elliptic Curve Digital Signature Algorithm, which is vulnerable to Shor's quantum algorithm. Therefore, we chose to apply quantum computer–resistant signature schemes. In general, we may use lattice-, multivariate-, and hash-based signature schemes or other alternatives. However, we recommend hash-based signature schemes for a few main reasons (although these primitives have more positive implications as presented later). First, hash-based digital signature schemes are solely based on the security of the underlying hash function. Thus, we can base the security of the whole blockchain protocol and the authenticity on one assumption rather than multiple assumptions, as the critical part to be protected is the blockchain itself. One could carefully state that the digital signature scheme remains secure as long as the blockchain is secure. Second, hash-based signature schemes have been investigated for a long time and are about to finish the standardization process. Furthermore, public keys are just hash values, which are space efficient and perfectly comply with the overall protocol.

Diversification via hash combiners. Due to possible advancements in cryptanalysis, it is conceivable that certain hash functions suddenly become insecure or at least do not satisfy all required security features, as can be seen by the attacks on MD5 or the more recent attack on SHA-1.

In such situations, one would like to have a mechanism that still ensures all desired properties without setting up a new blockchain protocol, which may involve a great deal of expense. Therefore, we propose a diversification strategy, in which we suggest applying hash combiners, which combine a number of hash functions such that the combination ensures all the security properties

as long as one of the hash function does. Hence, if one of the underlying hash functions is attacked, the security of the blockchain protocol is still preserved by use of the combiner.

Long-term security. In the long run, it may be possible, though expected with small probability, that in the worst case, all the applied hash functions within a blockchain become insecure. In some scenarios, one can restart with a secure instantiation of the protocol and just continue as if nothing happened. However, in other scenarios, the whole blockchain has to be rebuilt. This task is very inefficient as it involves interacting and communicating with network participants if the data is not available. Furthermore, the different instantiations such as PoW instead of PoS increase the burden on the system. Thus, minimizing the risk of such a situation by applying as many preemptive measures as possible is in our best interest.

> **If one of the underlying hash functions is attacked, the security of the blockchain protocol is still preserved by use of the combiner.**

Our Authenticated Blockchain Protocol—PQChain

Here, we propose our protocol, which uses the building blocks and design decisions we covered.

We mainly address the security of blockchain protocols, thus protecting against future threats and increasing their life cycle. This is attained by use of postquantum primitives such as efficient hash-based digital signature schemes. For legally binding purposes, we apply an identity-based verification mechanism, which we derive from existing protocols. Finally, we consider how to achieve a reactive blockchain protocol providing long-term security.

To this end, we exploit the blockchain protocol Certcoin[4] and the related algorithms in order to adapt them to the special case of hash-based signature schemes, which are stateful and support only a bounded number of signatures.

As already explained, we prefer hash-based signature schemes as they allow minimizing the number of security assumptions within the protocol. Furthermore, a public ledger B is now operated with a hash combiner Comb^{H_1,H_2}, combining two or more hash functions as opposed to previous constructions with one single hash function.

Stateful Digital Signature Schemes and Challenges

We briefly recap the notion of stateful signature schemes with public states. Applying such algorithms requires updating the state regularly to satisfy the fundamental security claims, whereas neglecting to update the state may involve major security loss or even allow breaking of the scheme. Representatives of such stateful signature schemes are the Gentry-Peikert-Vaikuntanathan (GPV) signature scheme[12] or hash-based ones. While the former only requires appending a uniform random string to the message in order to avoid maintaining the state, the latter leads to major loss of efficiency when applying stateless hash-based signature schemes instead.[13] A further challenge that we face is the fact that hash-based signature schemes such as plain XMSS[10] and XMSS-MT[14] can issue only a limited number of signatures. This could trivially be resolved by signing the root of a new tree, leading to an increased signature size, as the signature on the fresh public key has to be carried all the time. Suppose the state is lost and we do not want to maintain the state, or we wish to issue arbitrary many signatures without increasing the signature size. Can we use the public blockchain to tackle these challenges?

Stateful Signature Scheme

Formally, we define by M the message space and denote by $S = (S.\mathrm{KeyGen}, S.\mathrm{Sign}, S.\mathrm{Verify})$ the associated algorithms of a digital signature scheme S. These are defined as follows:

- $S.\mathrm{KeyGen}(1^\lambda)$: On input of the security parameter λ, the randomized algorithm outputs a public key pk, a secret key sk, and an initial state $state_0$.
- $S.\mathrm{Sign}(sk, state_i, m \in M)$: It is a randomized algorithm wherein on input of the secret key, the current state and the message outputs a signature σ. The state is updated to $state_{i+1}$.
- $S.\mathrm{Verify}(pk, m, \sigma)$: The deterministic algorithm outputs 1 if the signature is valid, otherwise 0.

Digital signature schemes usually satisfy existential unforgeability, meaning that it is infeasible for adversaries to generate a valid signature–message pair (m, σ) without the knowledge of the secret key sk, even when they see polynomially many message–signature pairs.

Protocol

Here, we show how to authenticate transactions within the blockchain and how to solve the challenges

described above. Our approach is mainly based on the Certcoin protocol[4] realizing a (decentralized) PKI, which is in turn based on the Namecoin protocol for (decentralized) DNS services.

Our goal is to merge the respective protocols with a transaction blockchain protocol such as Bitcoin in combination with an offline one-time authentication service in order to make transactions traceable to a certain natural person (not only an identity). The lack of such a functionality represents in some application scenarios one of the main drawbacks as institutions and service providers wish to have some legal protection or legally binding services.

To this end, we proceed in three steps. First, we show how to turn a stateful signature scheme into one in which the user is not required to maintain the state. In the second step, we use the blockchain to remove the bound on the number of signatures imposed by certain scheme instantiations such that the signer is free to issue arbitrary many signatures without lacking security. Finally, we show how to make the transactions traceable, in the sense that a legal institution can trace all the transactions related to an identity back to a natural person. This lays the foundations for legal protection.

Signature without State Maintenance

Let B denote the public blockchain ledger. Similar to Certcoin, we use the following procedures:

- B.register: Registers public keys related to identities.
- B.update$_i$: This procedure either updates the public key $(i = 0)$ related to an identity id or updates the state $(i = 1)$ related to a public key pk.
- B.lookup$_i$: For input $i = 0$ and id, it outputs the most current public key related to id. And for $i = 1$ and pk, it outputs the most current state related to pk.
- B.transaction: On input id, a transaction $<T>$, a signature σ_T on it, and a state $state_j$, it verifies the signature and that the transaction is admissible, updates the state, and puts the transaction on the ledger.

In our proposals, we can exploit accumulators or distributed hash tables as applied in Certcoin or other sophisticated protocols to realize efficient updates or lookups.

By $S_1=(S_1.\text{KeyGen}, S_1.\text{Sign}, S_1.\text{Verify})$, we define the set of algorithms of a signature scheme, where the state is maintained by the blockchain B. The description of the related algorithms is as follows:

- $S_1.\text{KeyGen}(1^\lambda) : (sk, pk, state_0)$: $\leftarrow S.\text{KeyGen}(1^\lambda)$.
- $S_1.\text{Sign}(\text{sk}, \text{pk}, <\text{T}>)$:
  1. $state_j \leftarrow B.\text{lookup}_1(pk)$
  2. $D = (<T>, state_{j+1})$
  3. $(\sigma_D, D) = S.\text{Sign}(sk, state_j, D)$
  4. B.transaction(id, σ_D, D)
- $S_1.\text{Verify}(pk, D, \sigma_D)$: $S.\text{Verify}(pk, D, \sigma_D)$.

We note that a signer can still choose to maintain the state without invoking blockchain lookups. Hybrid approaches are also possible. Clearly, the scheme remains secure assuming the blockchain protocol outputs always the most current state.

(Un)constrained number of signatures and identity-based verification. Because our goal is to operate the blockchain protocol in combination with hash-based signatures, where the number of leaf nodes is known beforehand, we discuss whether such constraints have a negative impact on the protocol. On the contrary, we highlight that by means of the public blockchain, one can exploit even smaller trees and thus make the protocol more efficient both in terms of signature size and performance due to the direct correspondence to the tree size. Intuitively, we can always update the public keys and store a chain of signed public keys within the blockchain such that a network participant can always trace back every used public key to the very first public key and hence the associated identity. Furthermore, because an identity is always coupled to an associated public key in the blockchain, we can verify each signature just by knowing the identity of the corresponding participant, similar to identity-based signature schemes. The protocol will be modified as follows: Denote by $S_2 = (S_2.\text{KeyGen}, S_2.\text{Sign}, S_2.\text{Verify})$ the modified signature scheme and B the public blockchain as above. Define by $\text{I}_{\max}$ the bound on the number of signatures.

- $S_2.\text{KeyGen}(1^\lambda)$:$(sk, pk, state_0) \leftarrow S.\text{KeyGen}(1^\lambda)$.
- $S_2.\text{Sign}(sk_{\text{current}}, pk_{\text{current}}, <T>)$:
  1. $state_j \leftarrow B.\text{lookup}_1(pk_{\text{current}})$
  2. **if** $(j < \text{I}_{\max} - 1)$
     - $D = (<T>, state_{j+1})$
     - $(\sigma_D, D) = S.\text{Sign}(sk_{\text{current}}, state_j, \text{D})$
     - B.transaction(id, σ_D, D)
  3. **else**
     - $sk^*, pk^*, state_0^* \leftarrow S.\text{KeyGen}(1^\lambda)$
     - $D^* = (pk^*, state_0^*)$
     - $(\sigma_{\text{pk}^*}, D^*) = S.\text{Sign}(sk_{\text{current}}, state_j, D^*)$
     - $B.\text{update}_0(\text{id}, \sigma_{\text{pk}^*}, D^*)$
     - $(sk_{\text{current}}, pk_{\text{current}}) \leftarrow (sk^*, pk^*)$
     - $D = (\langle T \rangle, state_1^*)$
     - $(\sigma_D, D) = S.\text{Sign}(sk_{\text{current}}, state_0^*, D)$
     - B.transaction(id, σ_D, D)
- $S_2.\text{Verify}(id, D, \sigma_D)$:
  1. $pk \leftarrow B.\text{lookup}_0(id)$
  2. **if** $(pk, D, \sigma_D) : S.\text{Verify}(pk, D, \sigma_D)$, **else**: output 0.

This protocol has several positive implications, which we summarize here. First, by using hash combiners for the ledger, we reduce the risk of future cryptanalysis attempts against certain hash functions. One may combine SHA3 with other finalists or second last round hash functions in order to diversify the portfolio. It is also possible to use the very same hash combiner (for instance, Comb^{H_1,H_2}) for the digital signature scheme.

Second, our choice to use hash-based signatures has, in a non-blockchain scenario, the disadvantage of allowing only for a bounded number of signatures. A trivial approach to solve this issue is to generate a new tree and sign the root with the old one. However, one always has to carry the chain of signatures in order to link them to the very first root. We mitigate this inefficient solution by use of the blockchain. Once an $(id, pk, state_j)$ tuple is verified by the network participants and subsequently added to the public ledger, the binding between the identity and the public key is updated. There is no need to carry the signature any more.

This has a very desirable impact on the efficiency of hash-based signature schemes since the signer is no longer encouraged to generate large trees. In fact, the signer can use smaller trees, directly translating into improved signature sizes and performance. In principle, one can choose a tree size consisting only of two leaf nodes. However, there will be some overhead related to the generation of a new tree and the blockchain update. Thus, one has to find the proper tradeoff.

> **Using a (decentralized) PKI allows realization of a signature scheme with identity-based verification, where the signer can choose the public secret key pairs.**

Finally, using a (decentralized) PKI allows realization of a signature scheme with identity-based verification, where the signer can choose the public secret key pairs. As opposed to identity-based signature schemes, no manager is involved, thus there is no key escrow and the signer is the only one who can sign on his or her behalf. This can be seen as an artifact of currently available (decentralized) PKI protocols. The verifier requires only the *id* of the signer and can, with the help of the blockchain, retrieve the most current public key (hash) associated to *id*. Such an identity-based scheme is a desirable feature if transactions have to be made traceable to an identity. This is especially interesting for legally binding protocols, contracts, transactions, and so on.

Tracing identities and privacy. Here, we very briefly discuss some ideas of how identities can be traced back to the associated persons. In some scenarios, the vendor or user wishes to have legal protection when triggering transactions. But this requires the involved parties to be identifiable—that is, transactions can be traced back directly to the vendor or buyer. However, the blockchain protocol in the current state does not come with such guarantees such that some central trusted authorities are indeed required, which couple the identity of a person with its name. Thus, the protocol has to be modified to the effect that blockchain participants accept new identities only if the identities and the associated public keys have been signed by a known trusted authority providing personal details on request. In particular, for private blockchains, the blockchain operators can define or set up trusted authorities.

For instance, if users desire to participate in the blockchain protocol, they may have to go to several trusted authorities to identify themselves, so security is ensured even if one or more trusted authorities have been compromised.

The ledger could also carry revocation lists so that the miners would refuse to accept transactions in case of compromise. The trusted authorities may also coordinate identification to avoid multiple identities coupled to one person. In other scenarios, a natural person may be allowed to take multiple identities for different reasons as long as he or she is registered with those identities at some known trusted authorities and can thus be traced. Such an approach, however, comes at the cost of sacrificing privacy and transaction anonymity since transactions can be linked to identities. Hybrid approaches are also conceivable, where the participant could operate in two modes within the blockchain: anonymous or identified using different public keys. The former restricts the participant's freedom to carry out certain transactions due to the requirement for traceability in case of a dispute. Thus, the participant must use its traceable keys for such transactions.

Security. Our approach allows update of current state-of-the-art blockchain protocols without affecting the respective security proofs, even though many do not come with a formal security analysis. Thus, if a blockchain protocol is proven secure in the old model and the hash function and signature scheme are replaced by (strongly robust) hash combiners and a hash-based signature scheme, respectively, satisfying the required

security notions, then the proof remains intact for the new protocol. The same also holds for the Certcoin protocol in combination with our changes. In fact, the only difference is that our protocol essentially stores a bit more information in the ledger than in the former approach. Furthermore, the protocol exploits identities to verify transactions because Certcoin keeps tuples of public keys (which are already hashes) and identities in the ledger.

Long-Term Security

For a blockchain protocol to function properly, the used signature scheme and hash function must be secure. If digital signatures or hash functions in the blockchain protocol become insecure, attackers may be able to alter existing transactions, or make transactions on behalf of other entities.

Threats

Typically used signature schemes and hash functions provide security under certain computational assumptions. If over time more powerful computers or algorithms become available, the security of these schemes degrades until eventually previously used cryptographic schemes are rendered insecure.

Consequently, it is necessary to prepare for replacing used cryptographic schemes with more secure ones in a protocol that is meant to run for decades or even longer.

The problem of weakened cryptographic primitives in blockchain protocols has been analyzed in "On Bitcoin Security in the Presence of Broken Cryptographic Primitives."[15] Furthermore, practical recommendations on what should be done in case used cryptographic primitives need to be replaced are given for the official Bitcoin protocol (https://en.bitcoin.it/wiki/Contingency_plans).

Countermeasures

Our blockchain protocol provides the following countermeasures against long-term security threats.

Minimal security assumptions. Our protocol enjoys minimal security assumptions, because its cryptographic security only relies on the security of the used hash function. In particular, any blockchain protocol must use at least a secure hash function to work properly.

Hash combiner. To make the hash function used in our protocol more robust against security failures, we use hash combiners, which again allow the combination of two hash functions (or more) such that the resulting hash value enjoys a relevant security property as long as one of the used hash functions enjoys this property.

For our blockchain protocol, we are interested in hash combiners that preserve the properties proof-of-work compatibility and collision resistance. Proof-of-work compatibility is required for a proof-of-work instantiation, which involves many tries prior to finding a proper η for given h_{-1} and m such that $H(h_{-1} \,||\, \eta \,||\, m)$ is smaller than a certain threshold. Collision resistance ensures unforgeability of hash-based signature schemes, preventing an adversary from finding distinct messages for the same signature. Interesting candidates for a hash combiner for our blockchain protocol are the combiners Comb_{4P} and $\text{Comb}_{4P\&OW}$ proposed by Fischlin and colleagues.[11] These combiners are both robust for collision resistance and pseudorandomness; the latter is even robust for one-wayness and can thus be used to improve the robustness of the hash-based signature scheme applied in our protocol. It is an interesting question if Comb_{4P}, for instance, also preserves proof-of-work compatibility.

Hash function replacement. Even when using a hash combiner, it may happen at some point that the provided security level is not sufficient anymore. In this case, our blockchain protocol allows replacing one or all of the currently used hash functions while keeping the blockchain intact. This works as follows:

- *Initialization of migration*: Suppose that hash combiner Comb^{H_1,H_2} is used and hash function H_2 is insecure. In this case, H_1 still ensures security and nothing has to be done. Alternatively, H_2 can be replaced. However, if H_1 is also threatened to become insecure in the foreseeable future, this requires the network participants to agree on a new hash function H_1^* (or combiner) and broadcast the usage of the new hash function over the network.
- *Migration procedure*: When a participant receives a migration message, he or she computes a hash h^* of the whole blockchain using the new hash function H_1^*. We stress that, in this procedure, not only the transactions in the previous block—but all transactions in the entire blockchain—must be rehashed. He or she then mines for a new special migration block with hash pointer h^* that also contains a description of the new hash function H_1^*. We stress that hashing the complete blockchain in this procedure is necessary to establish a consistent view of the current blockchain based on the security of the next hash function H_1^*. It is important that hash function renewal is done at a point in time when the old hash function H_1 still provides a sufficient level of security. Furthermore, we note that all the blockchain transactions may be included in one block in order to simplify the migration procedure.
- *Verification of old transactions*: For verification of a transaction using H_1, we must further check that the transaction and its signature are part of the view

established by the corresponding migration block. We must also check that H_1 was indeed considered secure when the migration block was generated. If both hold, then it is guaranteed that the transaction was already made at a time when H_1 was considered secure. Hence, it can be considered valid.

In this work, we highlighted the importance of a well-conceived strategy for an appropriate blockchain instantiation, as it represents a complex protocol with a huge number of participants, where trust in centralized authorities is replaced by the security guarantees provided by cryptography. Thus, it is necessary to ensure that the applied cryptographic primitives remain secure and stable preferably for the whole lifetime of the protocol, since an unexpected breakdown might have disastrous consequences. The blockchain protocol faces various threats imposed by powerful adversaries and we proposed a strategy including countermeasures against those threats. Investigating the required minimal security guarantees of the applied hash functions is left open as future work. ■

References

1. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.
2. P.W. Shor, "Polynominal Time Algorithms for Discrete Logarithms and Factoring on a Quantum Computer," *Proc. First International Symposium Algorithmic Number Theory* (ANTS-I), 1994, p. 289.
3. I. Eyal et al., "Bitcoin-ng: A Scalable Blockchain Protocol," *13th USENIX Symposium on Networked Systems Design and Implementation* (NSDI 16), USENIX Association, 2016, pp. 45–59.
4. S.Y. Conner Fromknecht and D. Velicanu, "A Decentralized Public Key Infrastructure with Identity Retention," Cryptology ePrint Archive, Report 2014/803, 2014; http://eprint.iacr.org/2014/803.
5. M. Ali et al., "Blockstack: A Global Naming and Storage System Secured by Blockchains," *USENIX ATC 16*, 2016, pp. 181–194.
6. P. Rogaway and T. Shrimpton, "Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance," *FSE*, 2004, pp. 371–388.
7. R. Pass, L. Seeman, and A. Shelat, "Analysis of the Blockchain Protocol in Asynchronous Networks," *EUROCRYPT*, 2017, pp. 643–673.
8. L. Lamport, R.E. Shostak, and M.C. Pease, "The Byzantine Generals Problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, 1982, pp. 382–401.
9. R.C. Merkle, "A Certified Digital Signature," *CRYPTO*, 1989, pp. 218–238.
10. J.A. Buchmann, E. Dahmen, and A. Hülsing, "XMSS—A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions," *PQCrypto*, 2011, pp. 117–129.
11. M. Fischlin, A. Lehmann, and K. Pietrzak, "Robust Multi-Property Combiners for Hash Functions," *J. Cryptology*, vol. 27, no. 3, 2014, pp. 397–428.
12. C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for Hard Lattices and New Cryptographic Constructions," *ACM Symposium on Theory of Computing*, 2008.
13. D.J. Bernstein et al., "SPHINCS: Practical Stateless Hash-Based Signatures," *EUROCRYPT*, 2015, pp. 368–397.
14. A. Hülsing, L. Rausch, and J.A. Buchmann, "Optimal Parameters for XMSS MT," *CD-ARES 2013 Workshops: MoCrySEn and SeCIHD*, Springer, 2013.
15. I. Giechaskiel, C.J.F. Cremers, and K.B. Rasmussen, "On Bitcoin Security in the Presence of Broken Cryptographic Primitives," *ESORICS*, 2016, pp. 201–222.

Rachid El Bansarkhani is a Postdoctoral Fellow at Technische Universität Darmstadt and the Founder, CEO of QuantiCor Security. He studied business computer science, mathematics, and received his PhD from TU Darmstadt in 2015. His research focuses on designing postquantum cryptographic schemes. He held several roles as Principal Investigator of various projects related to practical postquantum cryptography. Contact him at elbansarkhani@cdc.informatik.tu-darmstadt.de.

Matthias Geihs is a PhD candidate in the group "Cryptography and Computer Algebra" lead by Professor Johannes Buchmann at TU Darmstadt. He holds a Bachelor's degree in Computer Science with a minor in Mathematics from WWU Münster and a Master's degree in Computer Science with a minor in Business Administration from TU Darmstadt. His current research focuses on the design and security analysis of long-term secure systems. Contact him at mgeihs@cdc.informatik.tu-darmstadt.de.

Johannes Buchmann is a Professor of Computer Science and Mathematics, the Spokesperson of the Collaborative Research Center CROSSING of the German Research Foundation, the spokesperson of the Profile Area CYSEC at Technische Universität Darmstadt, and the Deputy Speaker of the Center for Research in Security and Privacy CRISP. Contact him at buchmann@cdc.informatik.tu-darmstadt.de.

Botnet in the Browser:

Understanding Threats Caused by Malicious Browser Extensions

Raffaello Perrotta and Feng Hao | Newcastle University

Browser extension systems risk exposing APIs, which are too permissive and cohesive with the browser's internal structure, leaving a hole for malicious developers to exploit security-critical functionality. We present a botnet framework based on malicious browser extensions and provide an exhaustive range of attacks that can be launched in this framework.

Attacks launched from within the browser have increased dramatically in recent years.[1] Although browsers have taken steps to restrict the installation of extensions to trusted sources, instances of malware being packaged into seemingly innocuous extensions are still regularly being reported. For example, a study conducted in 2014 by Kapravelos and colleagues discovered 130 compromised Chrome extensions.[2] In 2016, Malwarebytes identified a rogue extension[3] uploaded to Chrome's web store known as iCalc, which had bypassed Google's automated extension review audit and silently transmitted sensitive data to a rogue server. A recent study by researchers from Google highlights extensive efforts by criminals to distribute malicious extensions through Chrome's web store: nearly 10 percent of the reviewed Chrome extensions were identified as malicious between 2012 and 2015.[1]

Browser extensions are an attractive target for botnets, which are responsible for many large-scale attacks on the Internet infrastructure today.[4] First, they potentially have a large installation base with hundreds of millions of users, especially when an adversary opts to distribute malicious extensions by first buying an existing popular extension and then adding malicious code in the update.[1] Second, a browser extension possesses many over-privileged capabilities in accessing sensitive user data. Such capabilities can be easily abused. Finally, it is relatively easy to trick a user into installing an innocuous-looking extension that performs subtle malicious activities in the background.[5] Once installed, an extension forms an integral part of the browser and is outside the control of antivirus software installed on the user's computer.

In an early study about botnets in the browser, Liu and colleagues[4] proposed a botnet framework based on browser extensions. Their system relied on exploiting the extension update mechanism, which browsers provide, to issue commands in batch. The commands would then be processed by the installed extensions according to their designed format. Within their implementation, they include three attack examples: email spamming, distributed denial of service (DDoS), and password sniffing. Their architecture relies on the extension checking for updates using the extension's own update mechanism, which is only polled during events such as browser startup. This means an attacker cannot target commands to specific users in the network. Although there are follow-up papers by Liu and colleagues in 2012[6] and other researchers,[7,8] they are all limited in covering only a subset of botnet attacks. (For more information on malicious browser extensions, see the sidebar.)

 Copublished by the IEEE Computer and Reliability Societies

In this article, we aim to systematize the knowledge in this domain to raise the awareness of the threats present in modern browsers caused by malicious browser extensions. To this end, we present an extension-based botnet framework that allows fine-grained controls via a phone-home-based model, and an exhaustive range of attacks that can be launched by malicious browser extensions. The attacks are systematically categorized, described, and implemented on Chrome, Firefox, and Firefox for Android, which correspond to 59.24, 13.29, and 0.68 percent of the browser market share,[9] respectively. All the attacks have been experimentally validated against Chrome 54 and Firefox 49 (on both desktop and mobile)—which were the versions most recently available during the study—and on Windows, Linux, and Android systems. To the best of our knowledge, the results we present are the most comprehensive in the literature about the threats of botnet in modern browsers imposed by malicious browser extensions. Finally, we also discuss countermeasures to the identified problems.

Background

Modern browser extension systems are based on the JSE (JavaScript Engine) model,[10] in which the browser extension is a small set of scripts, generally JavaScript, running as part of the browser process. Extensions have evolved over time to reflect the needs of their user base, which has subsequently led to more privileged extension models being introduced. The majority of extension systems are free to read and manipulate the webpage's Document Object Model (DOM). In addition, many common privileges include access to the user's browsing history, bookmarks, and even lower-level functionality such as arbitrary file access. The levels of privilege vary depending on the security model set out by the browser's implementation. Chrome utilizes an extension system that heavily makes use of the isolated worlds paradigm, which in layman's terms simply means each script runs within its own sandbox. Firefox on the other hand, including Firefox for Android, utilizes an extension model that has been subject to wide criticism, with Liverani and Freeman claiming security was "almost inexistent."[11]

Many common privileges include access to the user's browsing history, bookmarks, and even lower-level functionality such as arbitrary file access.

Chrome Extensions

Chrome extensions comprise three distinct components: the manifest file, the event pages, and content scripts. Additional pages such as HTML or stylesheets may also be included within the extension, provided they are declared in the manifest. Chrome enforces, in its extensions, the principle of least authority (POLA), a motif in the security community that entails allocating only the permissions necessary to complete a specific action. A manifest file must be declared to specify which permissions and URLs the extension is allowed to access. In theory, this fine granularity of permissions should deter users from installing extensions that request permissions outside its intended scope. However, studies have shown that permission systems are not as effective as predicted[12,13] as most users tend to trust applications that seem to come from reputable or popular sources. Some users are also not aware of potential security implications, or simply do not find taking the risk of installing permissive applications to be problematic.

Event pages act as the intermediate layer between the browser and the content scripts. Event pages run as a separate process in Google Chrome and access browser contexts via APIs, which bridge the sensitive internal browser operations to a set of reduced calls that guarantee restriction in what the extensions themselves have the power to do. In the situation an event page wants to modify the page's content, communication may bilaterally occur between the event pages and content scripts; it is important to note that event pages do not have direct access to the DOM.

Content scripts are JavaScript files that are injected in a webpage and run in the same process space as the document that is rendered. They have the ability to read and manipulate the DOM without restrictions, provided the appropriate permission is adjudicated. Furthermore, content scripts may arbitrarily issue outbound cross-origin HTTP requests, without requesting additional permissions. This exposes a design decision that has some security implications to it, as we explain later.

Firefox Extensions

Jetpack extensions were introduced by Mozilla in 2013 as a solution to the archaic XPCOM-based extension system used by early versions of Firefox. XPCOM, which stands for cross-platform component object model, provides access to lower-level interfaces. In XPCOM, the functionalities allowed include reading system files and utilizing system libraries. Jetpack

was devised as a model that utilized the POLA within its modules to contain vulnerabilities. Jetpack extensions, in similar regard to Chrome, comprise a manifest file, at least one module (which mirrors event pages in Chrome), and content scripts. An interesting note is that the manifest file in Jetpack extensions specifies metadata rather than permissions. This is in contrast to Chrome, which employs a detailed permission schema and displays it to the user at install time.

Modules serve as the interface between native browser APIs and any content scripts that have been registered in the extension. Mozilla provides a set of core modules that can be reused with the aim to provide a safer interface to low-level functions. These core modules can be used indiscriminately by an extension developer. It is expected that in the Jetpack implementation, a strong isolation policy between modules and access to the native XPCOM components should be in place; however, even as of recently, Add-on SDK still allows developers to access the majority of the XPCOM functions. Although, like Chrome, modules do not have access to the DOM of a page; a similar message-passing system allows communication between the module APIs and a content script. Content scripts in Firefox perform a role identical to that described in Chrome's model.

Firefox for Android. Firefox for Android, also known as the Fennec project, was introduced in 2011. It is essentially a port of the main Firefox adapted to run on the Android platform. The code pertinent to our research, within the Add-on SDK, is nearly a one-to-one mapping between standard Firefox and its Fennec counterpart. Due to the differences between the security policies of the Android operating system compared to a PC (say Windows) operating system, we do observe differences in the impacts of some of our attacks. A similar observation can be made due to the introduction of more sensor devices, by default, on a standard smartphone. We later discuss in more detail the impact of attacks in different operating systems.

Threat Model

In our threat model, we assume a malicious extension has been installed on a target's system. There are many methods that an adversary can use to persuade a user to install a malicious browser extension, including disguising malicious extensions as legitimate browser extensions, using Trojans to install malicious extensions, and launching missing plugin attacks.[5] Another method available to attackers is to purchase a popular extension from its creator and then add malicious code to it. This technique has been observed to be used by adware producers.[1] Once a malicious extension is installed, we assume it has been granted all the privileges it needs to run, but it is constrained by the general sandboxing policies for browser extensions.

This threat model is different from the model commonly assumed by the browser vendors that browser extensions should always be "trusted" as long as they pass checks in a vetting process. We justify our threat model as follows. First, we should point out that the vetting process is never bulletproof, which is acknowledged by browser vendors. For example, researchers from Google Chrome published their experience and lessons from years of fighting malicious extensions.[1] They reported that by using advanced detection techniques for the vetting process, the best detection rate that their system was able to achieve is 96.5 percent. Second, we argue that a dedicated study on malicious extensions should be useful. Many users are not aware that a malicious extension can cause harm to their security and privacy. On the other hand, some security researchers might think of the other extreme: once a malicious extension is installed, all security is lost. In fact, malicious extensions are constrained by the sandboxing policies of the browser. The exact threats that a malicious extension can impose on users vary among browsers, the underlying extension architectures, and operating systems. The main aim in this article is to establish a comprehensive and up-to-date understanding on precisely what malicious extensions can and cannot do in modern browsers.

The threats we present in our model are the expressed potential for the concurrence of a harmful event that can breach confidentiality, integrity, and availability (CIA). To illustrate the threats, we implement a wide series of attacks that a malicious botmaster can launch by abusing the powerful APIs offered by the extensions. During the experiments, we also discover several inherent vulnerabilities, which make some browsers (for instance, Firefox) more susceptible to an attack than others (for instance, Chrome), which we detail later.

Botnet-Based Attack Framework

Botnets are prevalent among the criminal underworld of the Internet. They serve as a tool to facilitate attacks against numerous victims at the same time. Historically, botnets made use of the Internet Relay Chat (IRC) protocol to distribute commands to the compromised zombie computers. However, with IRC falling into gradual disuse, the medium through which a botnet receives its commands is changing. Some solutions use simple transport layer protocols such as TCP (Transmission Control Protocol) or UDP (User Datagram Protocol); however, due to its simplicity and ease of access at the application layer, HTTP has become more popular as the next transmission medium.[14]

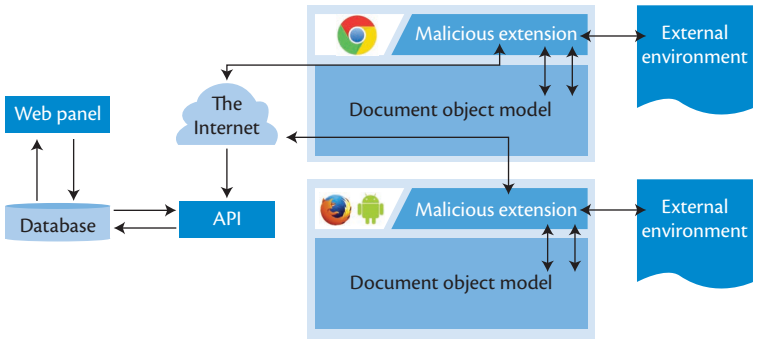


Figure 1. Our botnet-based framework.

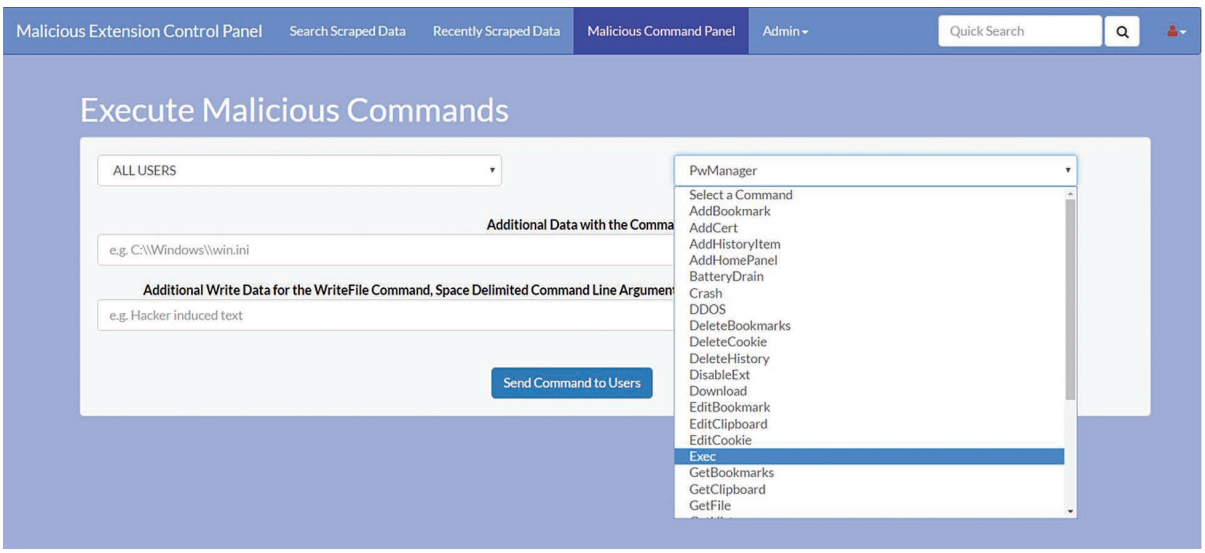


Figure 2. The web panel for the bot master.

The Framework's Architecture

We designed an architecture for complete control over a botnet of extensions, utilizing a simple Command & Control model (C&C).[15] We illustrate the effect of our attacks using this architecture on Chrome, Firefox, and Firefox for Android in our model. We illustrate in Figure 1 a component diagram representing an overview of our botnet system. Our framework provides the following functionality:

- An API schema that emphasizes independence from a target extension system, using inspiration from REST;
- Extensible command issuing system, along with data categorization utilities;
- An easy-to-use graphic user interface (see Figure 2); and
- Three examples of malicious extensions that exhaustively implement malicious capabilities respective to Chrome, Firefox, and Firefox for Android.

The Framework in Operation

Each extension uses a phone-home beacon periodically to identify itself to the botmaster's server. In our implementation, we randomly generate and store a unique identifier that corresponds to one instance of an extension. If the user uses social media (say Facebook), we further tie the victim's identity to their social media account. We note that several other methods exist in identifying a user, including via IP address or fingerprinting techniques using metadata available to the browser. One advantage of our botnet framework is that it allows automated testing on an exhaustive list of capabilities that a malicious extension may exploit. On receiving an identity, the botmaster's server will automatically find a list of commands associated with the identity and return it to the extension. The extension will automatically execute said command and mark it as executed. Victim data is then sent to the botmaster,

although we note that certain attacks operate passively. For example, HTML data is periodically sent to the remote server at a set interval.

Botnet Structures

We briefly discuss other common botnet paradigms and how feasibly they can be implemented using extensions.

Rallying techniques. Rallying refers to the process in which a C&C server discovers the domain/IP address of the central command server.[15] Hard-coding the domain name is the easiest solution; however, obfuscation techniques using minifiers are a viable alternative when using extensions. Other masking techniques include using Dynamic DNS services. Using this method, an attacker can mask the real C&C address behind a series of DNS servers, in which the path to the command server can be made resilient to mitigate against law enforcement shutting a server down. The use of domain generation algorithms can be deployed by including the generation algorithm in the source code of the extension.

Peer-to-peer botnets. Peer-to-peer (P2P) botnets are a phenomena that developed after the popularization of P2P networks. Each node acts as both a client and a server, distributing commands to each peer it has registered as well as executing them. Browser-based P2P botnets are made possible by a new technology called WebRTC, which provides APIs for P2P data interchange. WebRTC allows web browsers to request resources not only from backend servers but also directly from browsers of other users. Our experiments show that P2P botnets could be implemented using WebRTC on both browsers, which will make it harder for law enforcements to take down a botnet.

Threats Caused by Malicious Extensions

In this section, we systematically categorize, describe, and tabulate threats caused by malicious extensions, with an analysis against the CIA triad model. To illustrate the threats, we implement concrete attacks and verify them against Chrome, Firefox, and Firefox for Android browsers on multiple operating systems including Windows, Linux, and Android. Details of the attacks are presented below. See Table 1 for a summary of the results.

DOM-Based Capabilities

The DOM has been highlighted in a variety of prior studies[6–8,16,17] as a common source for malicious exploitation. Similarly, the near-unrestricted access to JavaScript functionality has also been discussed at length in previous studies.[2,18] Generally, these studies take the approach of exploiting benign-but-buggy extensions, rather than assuming the role of an attacker who has gained control over the extension's source code.

Full DOM access. We confirm the prevalent belief that extensions currently offer minimal protection against full DOM manipulation. The case is slightly mitigated by Chrome's permission systems restricting content scripts to executing on sites declared in the manifest file. However, this is impinged on by extension developers setting broad permissive tags that allow content script execution on any webpage.

Iframe-based phishing. Prior works have identified iframes as troublesome in making indirect cross-site requests[7] and as a prominent malvertising source.[19] However, we specifically observe their power in launching phishing attacks against HTTPS sites. Commonly, a user is instructed to check the URL of a website as well as the information of the green lock next to the URL bar to verify that the integrity of the site has not been compromised and no intruder is launching a man-in-the-middle attack. Our iframe attack was able to seamlessly substitute the page with an attacker's phishing site, while giving the victim the credence that they were still browsing their original site. This was done without tampering either with the URL bar or the information shown in the lock dialog. We motivate this attack with an example, shown in Figure 3, illustrating the Facebook page substituted with the phishing page: Phishbook.

> **Our iframe attack was able to seamlessly substitute the page with an attacker's phishing site, while giving the victim the credence that they were still browsing their original site.**

JavaScript-Based Capabilities

JavaScript brings a plethora of functionality that can be of interest to an attacker. Listeners for keystrokes, mousestrokes, and touchstrokes were found to be applicable in attacks against both browsers. Furthermore, attacks against the availability of the browser via executing resource-exhausting scripts were found to be effective, especially on Firefox, which has no recovery safeguard in place due to its monolithic process architecture. Furthermore, sensor-based APIs such as

Table 1. Summary of capabilities that malicious extensions likely exploit to compromise a user's security and privacy.

Capabilities category (botnet commands)	Chrome (54)	Firefox (49)	Firefox Android (49)	CIA triad impact	Comments
DOM-based capabilities					
Read webpage's DOM[6,7,17]	✓	✓	✓	Confidentiality	Full read access to a page's DOM is possible.
Edit webpage's DOM[6,7,16]	✓	✓	✓	Integrity, availability	Editing existing DOM elements is possible.
Write to webpage's DOM[6,7,16]	✓	✓	✓	Integrity, availability	Appending new content to the DOM is possible.
Replace webpage's DOM[6,7,16]	✓	✓	✓	Integrity, availability	Replacing an element of the DOM with another is possible.
Iframe-based phishing	✓	✓	✓	Integrity, availability	Changing a page's entire DOM structure to an external iframe, without modifying the URL, is possible for both HTTP and HTTPS sites.
JavaScript-based capabilities					
Crash browser	Partial	✓	✓	Availability	Chrome's multi-process system mitigates the attack to only the active tab. Firefox is subject to an entire browser crash.
Use of eval[2,7,16,18,20]	Partial	Partial	Partial	Integrity	Chrome sandboxes instances of eval, though it can be disabled by editing the Content-Security Policy. Firefox's review process adopts a harsh extension rejection mechanism for extensions using eval. Eval can be abused to remotely execute tailored JavaScript code.
XHR requests[2,7,16,18,21]	✓	✓	✓	Confidentiality	XHR requests can be used to transmit DOM content, or supply botnet commands.
Location data[17]	✓	✓	✓	Confidentiality	The request is made using JavaScript originating from a site, rather than the extension, so it is not unreasonable to assume that a user may click allow, allowing GPS coordinates to be scraped. If the site has originally requested it, then the decision is stored and is not requested again by the extension.
Keystrokes[7]	✓	✓	✓	Confidentiality	Keystrokes can be captured and transmitted with relative ease.

Continued

Table 1. Summary of capabilities that malicious extensions likely exploit to compromise a user's security and privacy (cont.).

Capabilities category (botnet commands)	Chrome (54)	Firefox (49)	Firefox Android (49)	CIA triad impact	Comments
Mousestrokes and touchstrokes[17]	✓	✓	✓	Confidentiality	Mousestrokes or touchstrokes on Android can impact on a user's privacy such as capturing websites that use graphical passwords.
Cookie capabilities					
Read cookies[7,16]	✓	✓	✓	Confidentiality	A full listing of cookies can be read, including HTTP-Only and Secure cookies.
Edit cookies[7,16]	✓	✓	✓	Integrity, availability	Cookies can be edited indiscriminately.
Delete cookies[7,16]	✓	✓	✓	Integrity, availability	Cookies can be deleted indiscriminately.
Clipboard capabilities					
Read clipboard	✓	✓	✓	Confidentiality	Clipboard data can be read in both Chrome and Firefox.
Modify clipboard	✓	✓	✓	Integrity	Clipboard can be modified via the use of JavaScript in all browsers.
Bookmark capabilities					
Read bookmarks[7]	✓	✓	✗	Confidentiality	A full listing of user bookmarks can be accessed. Fennec provides no native method to interact with bookmarks.
Add bookmarks[7]	✓	✓	✗	Integrity	Adding bookmarks, including folders, is possible, except in Fennec.
Edit bookmarks[7]		✓	✗	Integrity, availability	Current bookmarks can be traversed and updated, leading to malicious URL changes, except in Fennec.
Delete bookmarks	✓	✓	✗	Availability	Chrome imposes restrictions on removing and adding bookmarks in the root folder; however, the rest are deletable. Not possible on Fennec.
Browsing history capabilities					
Read history[7]	✓	✓	✗	Confidentiality	Access to a user's browsing history is possible, though not supported yet in Add-on SDK for Fennec.
Write to history	✓	✗	✗	Integrity	Firefox does not natively support writing entries to the browser history.

Table 1. Summary of capabilities that malicious extensions likely exploit to compromise a user's security and privacy (cont.).

Capabilities category (botnet commands)	Chrome (54)	Firefox (49)	Firefox Android (49)	CIA triad impact	Comments
Delete history	✓	✗	✗	Availability	Firefox does not natively support deleting entries from the browser history.
File system capabilities					
Directory listing[7,16,17]	✗	✓	✓	Confidentiality	Directory listing is available on Firefox; Android's file system restrictions prevent some directories from being listed.
Read files[7,16,17]	✗	✓	✓	Confidentiality	Reading files, including system files, is available, though Android's default root permissions apply.
Edit files[7,16,17]	✗	✓	✓	Integrity	If the user permission with which Firefox was started matches the file's access permissions, then editing is possible.
Delete files[7,16,17]	✗	✓	✓	Availability	If the user permission with which Firefox was started matches the file's access permissions, then deleting content is possible.
Add new folders[7,16,17]	Partial	✓	✓	Integrity	Permission withstanding, adding folders to the file system is possible in Firefox. In Chrome, only subfolders may be added to the already configured downloads directory.
Add new files[7,16,17]	Partial	✓	✓	Integrity	Permission withstanding, adding files to the file system is possible in Firefox. In Chrome, downloading files programmatically is plausible, but only to the configured downloads folder. Prompts are displayed on dangerous file downloads within Chrome.
Execute processes[7,16,17]	✗	✓	Partial	Confidentiality, integrity, availability	Executing files, including terminal commands, is possible in Firefox. Android's scope is limited to only executing APK files.

Continued

Table 1. Summary of capabilities that malicious extensions likely exploit to compromise a user's security and privacy (cont.).

Capabilities category (botnet commands)	Chrome (54)	Firefox (49)	Firefox Android (49)	CIA triad impact	Comments
Extension management capabilities					
Disable extensions[7]	✓	✗	✗	Availability	Can be done on Chrome silently; functionality was recently removed from Firefox.
Uninstall extensions[7]	Partial	✓	✓	Availability	Chrome requires user confirmation, whereas Firefox can uninstall extensions silently.
Other capabilities					
Proxy settings	✓	✓	✓	Confidentiality, integrity, availability	Browser proxy settings can be modified to point to an attacker-controlled proxy.
Browser preference[7]	✗	✓	✓	Integrity	Firefox exposes methods to change internal browser preferences.
DDoS[6,17,21]	✓	✓	✓	Availability	XHR requests are not rate limited, thus extensions can be used for flooding-based DDoS attacks. Depends on JavaScript capabilities.
Password manager[7,16,17]	✗	✓	✓	Confidentiality	If a user has not set a master password, plaintext passwords can be gathered silently. If the user has set a master password, then it will require entering it within the prompt.
XPCOM usage[7,11,17,22]	✗	✓	✓	Confidentiality, integrity, availability	Can still be used in Jetpack extensions, though Firefox will be deprecating XPCOM soon.
System library/API usage[7,11,17,22]	✗	✓	✓	Confidentiality, integrity, availability	Low-level APIs such as WinAPI, Linux OS libraries, and the JNI on Android can be bridged and accessed by Firefox extensions.
Battery drain[17]	✗	✗	✓	Availability	It can be made so that a phone will not automatically go to sleep when left unattended.
Certificate exceptions	✗	✓	✓	Confidentiality, integrity	We introduce an attack and describe in this article. Has a dependency on file system capabilities.

Figure 3. The result of an extension substituting Facebook's page with "Phishbook" in an iframe-based phishing attack. The browser continues to identify the page as Facebook (note the genuine https://www.facebook.com in the URL).

HTML5's geolocation were requested as part of the website rather than the extension; thus if a user accepts access to their GPS device, the extension can read their geolocation coordinates.

Unrestricted cross-site requests. We validate that the extension models we evaluated do not restrict cross-site requests made via XHR (XML HTTP requests). XHR requests are the most powerful tool in a botnet's arsenal. Remote connections can be made to an attacker-controlled server for harvested data to be transmitted back as well as for commands to be distributed across the botnet.

Cross-Site Scripting versus Extensions

Cross-site scripting (XSS) can be used to launch the attacks we have described in the previous sections as they are JavaScript based. An XSS attacker relies on finding a vulnerability in a site, where they can launch an XSS payload. An attacker using an extension can always launch the attack via the content script.

Cookie Capabilities

To mitigate the impact of XSS attacks, flags demarcating the cookie as being inaccessible by JavaScript are used, known as HTTPOnly flags. We find that extensions have access to more powerful cookie APIs that allow access to cookies even if they are marked as HTTPOnly, Session, or Secure. Browsers allow access by extensions to these extended features as extensions are seen as "trusted" once installed, whereas regular JavaScript is not. By periodically transmitting cookie values back to the attacker's remote server, session hijacking attacks may be performed with relative ease.

Clipboard, Bookmark, and History Capabilities

We validate the findings performed in "An Empirical Study of Dangerous Behaviors in Firefox Extensions"[7] on Firefox, and apply the analysis on Chrome and Fennec. It was found that in Fennec, Mozilla has not yet implemented any Add-On SDK method of accessing the clipboard, bookmark, or history, although a bypass for the clipboard was found on the Android system

utilizing XPCOM functions. On Chrome and Firefox, it was found that full access to a user's browsing history was possible. Modifying or deleting items from the history was a privilege granted only to Chrome, although it is possible to use file capabilities to emulate this attack on Firefox by modifying the places.sqlite file directly. Bookmarks were found to be creatable, readable, updatable, and deletable on both desktop browsers. The clipboard was found to be accessible and modifiable on all browsers. All three functions discussed in this section can be used to invade a target's privacy as well as inject content such as illicit material into the target's browser.

Firefox Exclusive Attacks

From the literature analyzed in combination with our own analysis of Firefox and Fennec, we observe that its less restrictive permission model allows us to access the external environment of the operating system itself, as Firefox provides access to it via a series of APIs. The original development philosophy of Mozilla extensions was to make their extensions as powerful as the browser, and although over a decade of security analysis has been performed on the Firefox system, which resulted in more restrictions being applied, we made some critical observations.

File system capabilities. Firefox's arbitrary file system access has been a longstanding concern among the extension community. We confirm that Firefox still allows extensions to access the file system, resulting in the possibility of arbitrary file reading, writing, or deleting. Modifying folders and manipulating existing files and directory data was also confirmed as possible, as well as executing files and commands. As there exists no data transmission barriers between reading a file and an XHR request, an attacker can read the contents of a file and transmit them back using the botnet framework we have introduced. Another scenario an attacker could launch using our framework involves writing a file to a directory, then launching a command to execute said file.

> **Expired and untrusted (such as self-signed) certificates, along with certificates containing URL mismatches, could be silently marked as trusted.**

It is important to note that these capabilities operate on the premise of privilege inheritance. Hence, if the browser is started by a user that does not have access to writing to a certain directory, then the operation is refused. Firefox for Android is most strongly affected by this situation, as the Android operating system restricts a large portion of the file system from being read, written, or executed for phones that are not rooted. A similar case can be argued for desktop operating systems; however, traditional operating systems like Windows allow fewer restrictions with executing and reading files, by default.

XPCOM and system library usage. XPCOM is a powerful native interface used in Firefox to provide endpoints for use by extensions, which allows access to native browser methods that are used in the browser's programming interface. These features remain exposed in Add-on SDK. We also make the observation that operating system libraries, such as the WinAPI on Windows or the Java Native Interface (JNI) glue library on Android, can be accessed with little restriction. This can result in an extension calling low-level routines exposed in the operating system user space, such as accessing process memory.

A note on Firefox for Android. Fennec's Add-on SDK port was generally designed to be backward compatible with desktop Firefox. We thus observed a vast similarity in the range of attacks that could be performed on Android users who use Firefox. There exists one main study done on the security of Firefox for Android conducted by Marston and colleagues[17] in 2014. The researchers identified certain malicious capabilities in the Add-on system as well as some predicted vulnerabilities in the future. We confirmed Marston and colleagues' section of the study that focused on malicious extensions and found that their results could be repeated. We also found that their predicted vulnerabilities, based on our experiments, are currently not possible.

After applying Marston and colleagues' study to our botnet framework, we applied the analysis we performed on Chrome and Firefox to Fennec and programmed the malicious functionality in a sample extension. We found that, apart from the differences we have highlighted in Table 1 and the prior sections, attacks using sensors were more plausible as mobile phones are far more likely to contain devices such as GPS in their hardware than a desktop computer. Furthermore, we also conducted an attack on the battery by preventing the phone from autolocking, which shows that denial-of-service attacks against power resources are also possible.

Certificate Exception Attack

One benefit of a systematic categorization of capabilities is that it enables us to discover how seemingly

innocuous APIs can interact with the underlying system in an unexpected way to compromise the user's security. As a demonstration of this, we present a certification exception attack that has not been reported before.

The security of our online transactions critically depends on the HTTPS protocol and on the management of the public key certificates. A public key certificate is issued by a trusted certificate authority (CA) and cryptographically binds the site's identity with other details such as an expiry date in a digital signature. Within Firefox, including Fennec, it is possible for a user to manually specify a certificate exception via the user interface provided, if a site uses an untrusted or erroneous certificate.

Our proposed attack removes this element of interaction from the victim and allows certificate exceptions to be added silently without the victim being privy to this knowledge. This is because Firefox insecurely stores certificate exceptions in a plaintext file located in the user profile in a file named cert override.txt. An attacker can manipulate this file so that new certificate exceptions could be added. This attack was reproducible on Firefox for Android. We emphasize that this vulnerability does not exist in Chrome, where certificate exceptions are stored on a per-session basis.

As with our previous attacks, we integrated this attack so that it was possible to specify a site and certificate to add, remotely, via the botnet framework. It was found that expired and untrusted (such as self-signed) certificates, along with certificates containing URL mismatches, could be silently marked as trusted using this attack.

This attack has several implications. Ordinary web users often know to check the padlock as a security sign of a website. The browser must warn the user if there is any abnormality in the certificate. However, our attack shows that the presence of a padlock has little meaning when the exception warnings can be silently suppressed by an extension. Furthermore, if a certificate exception is added without the user's knowledge, then the attack can be leveraged to perform man-in-the-middle attacks.[23] Other implications include marking outdated certificates as valid as well as allowing subdomain mismatches in certificates to occur, which can be a problem in shared domains.

Extension Management Capabilities

First analyzed by Wang and colleagues[7] on Firefox, browsers provide extension mechanisms to manipulate and access data regarding other extensions present on the browser. Capabilities that allow an extension to disable or uninstall are the most pertinent to an attacker, as this could lead to the removal of antimalware extensions from the victim's computer. Through our experiments, we found that Chrome did not allow silent uninstallation of other extensions, opting to show a dialog of confirmation instead. However, it did allow for extensions to be disabled silently without notifying the victim. On Firefox, a reverse of this situation was observed. Disabling an extension was no longer possible; however, uninstalling extensions was found to be achievable. This situation applied to Fennec as well.

Other Capabilities

As we highlight in Table 1, there are a variety of other problematic APIs that a malicious attacker may abuse. The proxy system was found to be editable for protocols such as HTTP and HTTPS on all three browsers examined.

This means that traffic could be routed to an attacker-controlled server and examined. Attacks against the availability of a user's Internet are also possible as an attacker can change the proxy address to a nonexistent IP address, causing connections to fail.

Firefox, including Fennec, exposes an API to the password manager system. These features were identified by Wang and colleagues,[7] and we verify them as still currently functional. By default, the password manager does not encrypt the password store with a master password until the user sets one up; this results in unrestricted access to passwords stored in the manager. If a user has set up a password manager, then the attack is still effective as long as the victim enters the password in the prompt offered by Firefox, and the extension will subsequently gain access to all the stored passwords.

Bypassing the Web Stores

We managed to upload a customized extension to Chrome's web store using the update system. The extension has the iframe-based phishing attack, which reads the command from our web server. We uploaded a similar extension to Mozilla's add-on store, based on the certificate exception attack. This extension also was designed so it could read commands from our web server. During the review in the vetting process, we disconnected the server so that no one would be affected by our (proof-of-concept) attacks.

To bypass the vetting process, we masked the malicious extension as a benign extension via the use of clean, structured, and well-commented code that indicated the extension served a valid research purpose. Although in the comments, we documented that the extension was written for a research project, we wanted to test if the extension could pass the automated checks by Chrome and Firefox. The vetting process by Chrome is primarily automated, and the extension we submitted there was quickly approved. Firefox has an additional stage of manual review, which took longer. But in the

end, the extension was also approved. We removed both extensions as soon as they passed the vetting process before anyone else was able to download them, following the guidelines from our university ethics committee.

We disclosed our findings to Google and Mozilla. Google replied by highlighting their web store policy regarding that they remove malicious extensions. However, our experiments indicated that uploading a malicious extension controlled by a botnet was possible. Mozilla replied by highlighting that they were changing the Firefox add-on development framework by moving toward WebExtension APIs.

Comparison with Malware

Malware is a term that aggregates and identifies programs that exhibit malicious behaviors toward their victims. Compared to traditional malware, an extension-based botnet has the advantage that administrator privileges are not required for installing the extension. Furthermore, once installed, extensions effectively form an integral part of the browser and hence can easily avoid detection by antivirus software. In this section, we compare our extension-based botnet to other malware according to the taxonomy of malware behaviors presented by Gregio and colleagues.[24]

Viruses, worms, and Trojans. Viruses can be produced via extensions but are more effective on Firefox due to its capability to write and execute files. Traditional worms face some restrictions in propagating via extensions. A classical worm will self-propagate, typically via a known exploit that requires little to no user interaction to spread the worm. To successfully launch a browser worm, exploits would need to be found in the browser's core logic. With this in mind, an adversary may still distribute worms via Firefox's extension system, due to its control of low-level APIs. To give a contemporary example, the ransomware WannaCry exploited a Windows flaw,[25] and then spread itself automatically using worm-like behavior by abusing the flaw in unpatched systems. With Firefox's ability to make low-level API calls, it's not too hard to envision the possibility of loading worms like WannaCry into an extension. However, this method would not be currently possible on Chrome. Finally, Trojans can be created with relative ease, and such was the case documented by Utakrit[26] in his analysis of extension-based banking Trojans.

Botnets and DDoS. We refer the reader to the Botnet-Based Attack Framework section for a detailed analysis of botnets and different architectures. Botnets are often used to launch DDoS attacks. To assess the feasibility, we implemented a DDoS attack based on the use of XHR for both extension systems.[21] We observed that an average of 252 XHR requests could be sent per second, from one machine. With some established botnets having reached over 180,000,[27] this number could be amplified to a maximum 45,360,000 requests per second. With such a number of requests, flooding-based DDoS becomes a possible attack that can be performed by extension botnets.

Spyware, adware, and ransomware. Table 1 documents a large portion of features that can be used to break the user's confidentiality. For example, with liberal access to the DOM possible, spyware can be written to simply transmit the user's browsing contents to the central command server on all three targeted browsers. Cookies, browsing history, bookmarks, keystrokes from within the browser focus, location data, and passwords can all be captured by extension-based spyware. In addition, adware can be introduced by breaking the integrity of the browser or DOM. In our implementation, we provide a spam attack that enables the injection of an attacker-defined HTML ad into the browser's current page. Ransomware cannot be effectively created in Chrome extensions, as it could only encrypt the contents of a webpage. In Firefox, ransomware could be created to encrypt some of the file system's contents, leaving victims unable to access their documents without paying a ransom for the decryption key.

Rootkits. Rootkits, by their traditional definition, give the attacker a way of acquiring root-or system-level permissions as a result of an exploit. This type of malware would be unlikely to deploy using extensions, especially in Chrome, due to its isolated worlds model, and Fennec as Android typically blocks root access by default. Such an attack would require a major vulnerability in the browser. Another possibility is that a Firefox extension can download and write a rootkit to the file system and execute it. However, this would still require the user to confirm the escalation of privileges dialog, which can raise the alert.

Surreptitious software. Code can be obfuscated in extensions by using a JavaScript minifier or by remotely loading the extension code from an attacker-controlled source. Obfuscation is commonly used to bypass Chrome's extension review process, whereas Firefox's review process can be bypassed by writing code that appears benign and utilizes social engineering to mislead the reviewers of its benignity. In the extension we uploaded to the Firefox add-on store, we heavily made use of comments describing to the reviewers that the extension was for internal use within our university only.

Related Work in Malicious Browser Extensions

Although the threats of malicious extensions have been reported before, we are not aware of any systematization of knowledge about effects of malicious extensions against major browsers across multiple operating systems. Our article contributes to this subject by systematically analyzing existing literature as well as running experiments and enumerating an exhaustive range of APIs that can be abused for malicious purposes.

Malicious Browser Extensions

Prior work was conducted in this area on the Google Chrome browser by Liu and colleagues,[6] who demonstrated the power that could be exploited from Chrome extensions as part of a rudimentary botnet. Wang and colleagues performed an empirical study of dangerous behaviors on Firefox extensions,[7] in which they examined a set of 2,465 web store extensions against an array of security-critical functionality of varying threat levels. Ter Louw and colleagues detailed a comprehensive browser extension study in 2008;[22] however, it largely focuses on the XPCOM architecture. We also note the works done by Acker and colleagues,[28] who identify malicious and vulnerable scripts in the popular Firefox community scripting extension, Greasemonkey.

Exploiting Benign Extensions

Works were done by Barth and colleagues[16] in designing the extension model that Chrome adopted. Their paper proposes an adapted version of their research to Firefox. Carlini and colleagues[18] evaluated the Chrome extension architecture for features that would bypass or compromise the principles proposed by Barth and colleagues. Following that analysis, they produced an evaluation of 100 randomly selected Chrome extensions to audit them for hidden vulnerabilities and discovered at least 40 percent had at least one vulnerability that could be exploited.

Extension Vulnerabilities

Generally, vulnerability exploitation is reported via Chrome and Firefox's respective vulnerability programs. Publishing an article about the vulnerability is left at the discretion of the reporter. However, there is adequate academic research targeted toward exploiting vulnerabilities within extensions themselves. Liverani and Freeman[11] further demonstrate XSS attacks against Firefox's chrome privilege zone.

Static and Dynamic Analysis Techniques

Static analysis techniques were introduced by Bandhakavi and colleagues,[20] creating the vetting tool VEX. The tool identifies potentially malicious flows in Firefox extensions by analyzing the source code. Dynamic analysis tools were introduced by studies such as those conducted by Dhawan and Ganapathy[10] and Kapravelos and colleagues.[2] These tools are built into the browser and involve tagging and monitoring objects that have an untrusted source as its provenance. Both of these analysis techniques can be used to diminish attempts to perform malicious actions successfully, although it is unknown how serious the impact of code obfuscation is.

Countermeasures

Having presented a thorough overview of the threats present in modern extensions, we now take a step back and discuss a range of countermeasures. Some of the threats/vulnerabilities identified in this article can be easily addressed, but for many others, fundamental changes in the underlying extension architecture are needed, which requires further research.

Restricting Arbitrary Access to the DOM

Liu and colleagues' paper proposed an extension of the privilege management system on Chrome.[6] They introduce the concept of sensitivity as a default protection mechanism against DOM elements that typically contain confidential information, such as password inputs. Sensitivity levels are by default applied to HTML elements according to a default mapping between levels and elements. For example, high sensitivity may be applied to input tags containing the password attribute, whereas medium sensitivity is applied to tags containing IDs or usernames. By default, other elements are marked as having a low sensitivity. Content scripts are marked with a default sensitivity level. This level corresponds to the elements from which they may read a value. If a content script has lower privileges than the sensitivity of the element it is attempting to read, the read will be rejected.

Iframe-Based Phishing

Using an extension to substitute or modify a page with an iframe that loads external content, especially on an HTTPS site, should be detected. This would add negligible overhead, as it would only require checking if an iframe is being inserted by a content script. For further usability, a user may be presented a confirmation dialog clearly illustrating the domain where the external content is loaded from, which they may then choose to allow or disallow. As this would apply only to iframes inserted by extensions, this would not affect advertising or legitimate web uses of iframes.

Defeating the Botnet: Restricting Cross-Site Requests

Cross-site requests are a dangerous tool as they allow sensitive information to be transmitted to a remote location. Proposals to curb the damage done by cross-site requests have been introduced by several studies. Liu and colleagues' aforementioned paper introduced a micro-privilege management schema that separates cross-site access between the extension core and the content scripts.[6] The extension core is granted explicit cross-site access privileges via the manifest file, whereas any attempt to bypass this via the use of a malicious content script is prevented by having it explicitly declare any remote origins added to the webpage. This schema informs the user which sites the extension has access to in transmitting information.

Over-Permissive Browser APIs

The main countermeasure used against the arbitrary use of JavaScript APIs is permission management. This method is only in use in Chrome, as it is necessary to explicitly declare in the manifest file the APIs that the extension uses. However, permission systems are not a silver bullet to preventing the malicious use of APIs. Application permission systems experience multiple shortcomings, with adware developers using enticement techniques and lookalike naming schemes to push more invasive permissions out to users.[12] Furthermore, additional evidence exists that permissions become less effective over an installed application's lifetime.

We recommend some minor changes to Chrome's permission system by marking more dangerous permissions with greater prominence on the installation prompt, rather than presenting them equally as if they have analogous security implications. This is an idea introduced by Felt and colleagues.[13] Ideally, no read data from browser APIs should be transmitted via a cross-site request, but enforcing this would not be practical by current technology. We therefore take a moderate stance in suggesting a combination of the countermeasures suggested in managing cross-site requests, alongside a well-designed permission system. We suggest in particular that Mozilla should follow this schema in WebExtensions.

Firefox and Fennec's Privileged Extension Model

Currently, Mozilla is attempting to phase in WebExtensions, a modernization of the previous extension systems. From recent analysis, it appears Mozilla has removed APIs that allowed access to the file system and other contexts outside the browser. However, their success will rely on the extension developers to migrate to using the proposed WebExtensions kit. There is no information indicating whether Jetpack will be fully removed or operate in parallel with WebExtensions.

Firefox's Certificate Exception Attack

Certificate exceptions should not be added by extensions nor stored in plaintext. The cert override.txt file poses a suitable target for malware running with non-superuser privileges. We recommend that the method of providing certificate exceptions should follow a system similar to that of Chrome, where certificate exceptions are always approved by the user on a per-session basis.

We have introduced a browser extension–based botnet framework, which we implemented as a way of demonstrating the impact of running botnets in Chrome, Firefox, and Firefox for Android Jetpack extensions. We enumerate the list of malicious capabilities of these extensions and discuss countermeasures to the identified security problems. ■

Acknowledgments

All of our experiments were approved by the university ethics committee. The proof-of-concept attacks were performed on the authors' own computers and used freshly created Facebook accounts for testing only. For obvious reasons, we cannot release the developed extensions as open source code, but we intend to make the code available, upon request, to researchers working in this field. Feng Hao would like to acknowledge the support of ERC Starting Grant 306994.

References

1. N. Jagpal et al., "Trends and Lessons from Three Years Fighting Malicious Extensions," *24th USENIX Security Symposium* (USENIX Security 15), 2015, pp. 579–593.
2. A. Kapravelo et al., "Hulk: Eliciting Malicious Behavior in Browser Extensions," *23rd USENIX Security Symposium* (USENIX Security 14), 2014, pp. 641–654.

3. J. Segura, "Rogue Google Chrome Extension Spies on You," Malwarebytes Lab, 26 Jan. 2016; https://blog.malwarebytes.org/online-security/2016/01/rogue-google-chrome-extension-spies-on-you.
4. L. Liu, X. Zhang, and S. Chen, "Botnet with Browser Extensions," *IEEE Third International Conference on Privacy, Security, Risk and Trust* (PASSAT 11) and *2011 IEEE Third International Conference on Social Computing* (SocialCom 11), 2011, pp. 1089–1094.
5. A. Saini, M.S. Gaur, and V. Laxmi, "The Darker Side of Firefox Extension," *Proceedings of the 6th International Conference on Security of Information and Networks*, ACM, 2013, pp. 316–320.
6. L. Liu et al., "Chrome Extensions: Threat Analysis and Countermeasures," Network and Distributed System Security Symposium (NDSS), 2012.
7. J. Wang et al., "An Empirical Study of Dangerous Behaviors in Firefox Extensions," *International Conference on Information Security*, Springer, 2012, pp. 188–203.
8. N. Golubovic, "Attacking Browser Extensions," Ruhr-Universitat Bochum, 3 May 2016; https://golubovic.net/thesis/master.pdf.
9. "Mobile/Tablet Browser Market Share," NetMarketShare; 15 Nov. 2016, https://www.netmarketshare.com/browser-market-share.aspx? qprid=0n&qpcustomd=1.
10. M. Dhawan and V. Ganapathy, "Analyzing Information Flow in JavaScript-Based Browser Extensions," *IEEE Computer Security Applications Conference*, 2009, pp. 382–391.
11. R.S. Liverani and N. Freeman, "Abusing Firefox Extensions," Defcon17, July 2009.
12. P.H. Chia, Y. Yamamoto, and N. Asokan, "Is This App Safe?: A Large Scale Study on Application Permissions and Risk Signals," *Proceedings of the 21st International Conference on World Wide Web*, ACM, 2012, pp. 311–320.
13. A.P. Felt, K. Greenwood, and D. Wagner, "The Effectiveness of Application Permissions," *Proceedings of the 2nd USENIX Conference on Web Application Development*, 2011, p. 7.
14. J.-S. Lee et al., "The Activity Analysis of Malicious HTTP-Based Botnets Using Degree of Periodic Repeatability," *Proc. IEEE Int'l Conf. Security Technology*, 2008, pp. 83–86.
15. S. Khattak et al., "A Taxonomy of Botnet Behavior, Detection, and Defense," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, 2014, pp. 898–924.
16. A. Barth et al., "Protecting Browsers from Extension Vulnerabilities," Network and Distributed System Security Symposium (NDSS), 2010.
17. J. Marston, K. Weldemariam, and M. Zulkernine, "On Evaluating and Securing Firefox for Android Bowser Extensions," *Proceedings of the 1st International Conference on Mobile Software Engineering and Systems*, 2014, pp. 27–36.
18. N. Carlini, A.P. Felt, and D. Wagner, "An Evaluation of the Google Chrome Extension Security Architecture," *21st USENIX Security Symposium* (USENIX Security 12), 2012, pp. 97–111.
19. X. Xing et al., "Understanding Malvertising through Ad-Injecting Browser Extensions," *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 1286–1295.
20. S. Bandhakavi et al., "Vetting Browser Extensions for Security Vulnerabilities with VEX," *Communications of the ACM*, vol. 54, no. 9, 2011, pp. 91–99.
21. G. Pellegrino et al., "Cashing out the Great Cannon? On Browser-Based DDoS Attacks and Economics," USENIX Workshop on Offensive Technologies (WOOT), 2015.
22. M. Ter Louw, J.S. Lim, and V.N. Venkatakrishnan, "Enhancing Web Browser Security against Malware Extensions," *Journal in Computer Virology*, vol. 4, no. 3, 2008, pp. 179–195.
23. F. Callegati, W. Cerroni, and M. Ramilli, "Man-in-the-Middle Attack to the HTTPS Protocol," *IEEE Security & Privacy*, vol. 7, no. 1, 2009, pp. 78–81.
24. A.R.A. Gregio et al., "Toward a Taxonomy of Malware Behaviors," *The Computer Journal*, vol. 58, no. 10, 2015, pp. 2758–2777.
25. "Ransom.Wannacry," Symantec, 12 May 2017; https://www.symantec.com/security_response/writeup.jsp?docid=2017-051310-3522-99.
26. N. Utakrit, "Review of Browser Extensions, a Man-in-the-Browser Phishing Techniques Targeting Bank Customers," *Proceedings of the 7th Australian Information Security Management Conference*, 2009; http://ro.ecu.edu.au/ism/19/.
27. B. Stone-Gross et al., "Your Botnet Is My Botnet: Analysis of a Botnet Takeover," *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ACM, 2009, pp. 635–647.
28. S. Van Acker et al., "Monkey-in-the-Browser: Malware and Vulnerabilities in Augmented Browsing Script Markets," *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, 2014, pp. 525–530.

Raffaello Perrotta is a PhD candidate in the Secure and Resilient Systems group, within the School of Computing Science, at Newcastle University. He received his BSc in computer science with first-class honours from Newcastle University. His area of interests include browser and web security, IoT security, and fuzzy key exchange. Contact at r.perrotta@newcastle.ac.uk.

Feng Hao received a PhD in computer science from the University of Cambridge in 2007. After working in security industry for several years, he joined the School of Computing Science, Newcastle University, as a lecturer in 2010. He is currently a reader (associate professor) in security engineering. He is a senior member of the IEEE. Contact at feng.hao@newcastle.ac.uk.

Editors: Heather Richter Lipford, richter@uncc.edu | Jessica Staddon, jessica.staddon@gmail.com

Inclusive Security and Privacy

Yang Wang | Syracuse University

Alex is blind and is in his 50s. He works as a librarian in a local public library. Part of his job is to help people with disabilities use library computers equipped with assistive technologies. Like many other people with visual impairments, Alex uses computers and browses the Internet via screen readers, a type of software that parses and reads aloud content displayed on a computer/device screen.

One day Alex was logging into Gmail on a library computer with the JAWS screen reader, and it took him more than five minutes to successfully log in. This was a frustrating experience not because he was unaware but because Gmail fell short of supporting people with visual impairments. In this particular case, Alex typed gmail.com in the browser address bar but could not find the field to type in his account name on the Gmail page after several attempts. Being an advanced user, he hypothesized that another user might have previously logged into Gmail on this computer. Next, he needed to confirm this hypothesis by identifying the name of the other user and finally did so after frustratingly combing his way through the page: *"OK, there it is…so that's her email."* After finding the name of the other user, he then struggled to find the button he needed to log into his own account because he was unsure of the terminology used to describe the login area: *"sometimes it's 'log in as another user,' sometimes it's 'sign in as another user,' sometimes it's 'change user.'"* He felt that constantly changing the terminology of login elements introduces a new and steep learning curve regarding how to locate the authentication area quickly and efficiently: *"unfortunately, this is something that we run into a lot, you don't know what they call things, and every time they update the website, you have to re-learn how to do it."* Alex eventually worked his way through the links to find the login page that asked him for his account name and password. He typed those in, and he was logged in. However, it was not readily clear to him that he had logged in and he had to go through the page to make that inference.

This was one of many challenging situations that we observed during our study on authentication experiences of people with visual impairments.[1] Alex is an advanced computer user, but even for him, a seemingly mundane authentication task can be both time consuming and error prone. He told us that many of his library patrons with disabilities were frustrated with current designs of computers and the Internet. What's worse, they blamed themselves rather than the technologies and gave up using the technologies.

Unfortunately, this is just one example of the exclusion of users with disabilities. This is not an issue that only people with visual impairments or disabilities experience. This is an example of much broader, complex, and systematic issues associated with today's end-user privacy and security designs. The root problem is that our user-facing privacy and security designs have not paid

Copublished by the IEEE Computer and Reliability Societies

enough attention to a wide range of under-studied users, such as children, older adults, people with disabilities, activists, journalists, victims of crimes or domestic violence, and people from non-Western or developing countries.

The security and privacy research community has made great strides in identifying security and privacy risks in information and communication technologies and designing various basic and applied countermeasures such as cryptography, encryption, access control, formal methods, secure computation, and privacy-preserving/enhancing techniques. The importance of the human aspects of privacy/security has also long been recognized. Since Saltzer and Schroeder's seminal work in 1975 advocating for computer security mechanisms to be "psychologically acceptable,"[2] the human factors and more specifically the usability of security and privacy mechanisms have become a key research topic (see *Usable Security: History, Themes, and Challenges*[3] for a recent review). For instance, Whitten and Tyler conducted a well-known user study of Pretty Good Privacy (PGP), an encryption program, and found that it was difficult for ordinary people to use it. Thus, this study exposed the limited value of PGP and highlighted the importance of usability in security mechanisms. More broadly, usability has been considered a first-class design requirement for security and privacy designs.[4] For the past decade, the community of usable privacy and security (for instance, the annual Symposium on Usable Privacy and Security [SOUPS]) has been growing. However, something is worth noting inside and outside of that community.

In the field of psychology, an influential meta-study found that over 80 percent of published psychological studies focused on people from Western, Educated, Industrialized, Educated, Rich, and Democratic (WEIRD) countries, and thus it is highly questionable whether the results can be generalized to people in other non-WEIRD countries.[5]

Unfortunately, I believe that a similar problem of not paying enough attention to the wide variety of user populations exists in the current usable privacy and security literature. Looking through the 13 years (2005–2017) of SOUPS conference proceedings, less than 10 percent of papers (about 20 out of 215 papers) had data about under-studied users (for instance, children, older adults, or people from non-WEIRD countries). This suggests that even though the privacy and security community has taken the human perspective seriously, it has narrowly focused on certain types of users. This is problematic because these under-studied user populations are essentially left out, and the current end-user privacy/security mechanisms fall short of supporting them to ensure their privacy and security.

Most security and privacy mechanisms were designed with the general population in mind, leaving many specific user groups under-studied and under-served, such as people with disabilities.

The good news is that some privacy/security research has started to pay attention to these under-studied users. For instance, UniPass is an accessible password manager for blind users,[6] and DigiSwitch is a device for older adults to monitor and control the collection and transmission of their health information.[7] These examples illustrate the prospect of making security and privacy designs more inclusive to under-served populations. However, these research efforts, while very valuable, are still in the periphery of the field. The wide range of under-served populations deserve more attention and research in a more systematic way.

Inclusive security and privacy (inclusive S&P)—the idea of designing security and privacy mechanisms that are inclusive to different human abilities, characteristics, needs, identities, and values—is a perspective that takes human abilities and characteristics as first-class design requirements and aims to design mechanisms that are inclusive to the widest possible range of users. Inclusive S&P needs a stronger presence to make it more mainstream, similar to the way usability was elevated and is now widely recognized in the field of security and privacy. Thus, inclusiveness should be expected in all security and privacy designs rather than being a property that only a few system designers espouse.

This article highlights the limitations of the current framings/foci of security and privacy research and advocates for a new perspective of security and privacy research.

A Research Framework of Inclusive Security and Privacy

The long-term research agenda of inclusive S&P is to design security and privacy mechanisms for everyone. This ambitious vision goes beyond making security and privacy designs usable. In addition to usability, inclusive S&P designs encompass different human abilities, characteristics, values, and needs.

There are three key insights that guide this new research perspective. First, most security and privacy

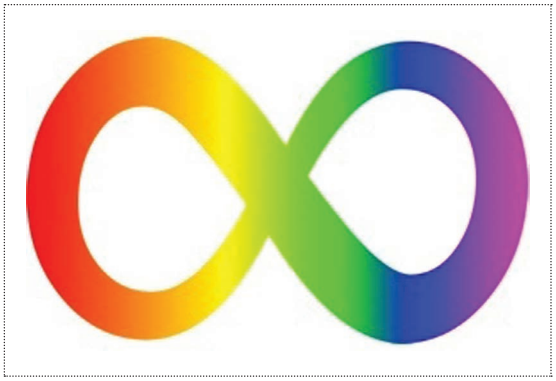


Figure 1. The neurodiversity sign symbolizes the diversity of human abilities, needs, characteristics, and values.

mechanisms were designed with the general population in mind, leaving many specific user groups under-studied and under-served, such as people with disabilities. Second, studying these under-served populations' security and privacy practices will not only deepen our understanding of their needs and challenges but also create an opportunity to examine and rethink more broadly about current security and privacy conceptualizations, methodologies, and designs. Third, designing for these under-served populations will not only create security and privacy mechanisms that better support them but also potentially benefit everyone—an embodiment of *universal design*.[8]

Research Challenges

The vision of universal design is design for everyone. In practice, this is very difficult if not impossible. Several empirical studies investigate the privacy/security concerns and practices of certain under-served populations such as children, older adults, people with disabilities, and people from non-Western or developing countries (see "The Third Wave?: Inclusive Privacy and Security"[9] for a recent review). For instance, Ahmed and colleagues interviewed people with visual impairments and found that they face difficulties in detecting visual or aural eavesdropping, have physical security and privacy concerns (for instance, using an ATM), and sometimes need to ask others (even strangers) to help (for instance when reading documents or typing in a PIN while shopping).[10] There are very few studies that focus on multiple under-served populations (for an example, see "An Inclusive, Value Sensitive Design Perspective on Future Identity Technologies"[11]).

People from different under-served groups may have profoundly different needs and challenges for security and privacy (Figure 1). In fact, even people with the same disability condition can vary significantly in terms of their abilities, needs, and technology uses. The scholarship from black feminist theories has proposed *intersectionality*, the idea that every person has a multifaceted identity consisting of race, class, gender, and sexuality.[12] This body of literature warns against over-generalization and advocates paying attention to individuals' complex identity structures and lived experiences.

Ethical challenges also exist. Some of these under-served populations may be considered vulnerable (for instance, children) and thus require researchers/designers to be extra cautious about how to preserve these users' interests. When working with under-served populations, researchers/designers might subconsciously bring their own biases especially when they are not part of the under-served groups. Feminist scholars have proposed the notion of *positionality*,[13] highlighting that research/design process is power laden, and urge researchers/designers to examine and mitigate their own biases. It is also worth noting that under-served populations may experience improvements of life during a study (for instance, trying out a research prototype), but they are likely to revert back to their previous life conditions after the study, which can be frustrating to say the least. Therefore, it is important for researchers to be mindful about addressing this challenge. For instance, the researchers may consider providing their participants the option of keeping the prototype after the study.

One reoccurring theme across many of these populations is people's pursuit of different (sometimes competing) values. Inclusive S&P designs need to consider the broader everyday context in which privacy and security are just two such values that people desire and might have to trade for other values (such as trust), depending on the situation.

Design for Inclusive Security and Privacy

The design for inclusive security and privacy can build on several lines of research, such as value-sensitive design (VSD) and accessible design. VSD is a generic design approach that highlights and supports values in system design such as user autonomy, freedom from bias, privacy, and trust.[14] It has been applied to assess technologies or privacy designs. For instance, Briggs and Thomas conducted workshops to understand people's perceptions of future identity technologies with six marginalized community groups: young people, older adults, refugees, black minority ethnic women, people with disabilities, and mental health service users.[11] They identified both common values and different impacting factors across these community groups regarding how people think about future identity technologies.[11] As shown in this example, VSD can be useful in identifying the underlying values that under-served user groups have and assessing whether these values have been supported in security and privacy designs. Any design has embedded values either explicitly

or implicitly. I advocate that inclusiveness is desirable, which is itself a value. Security/privacy designers need to make their value judgments and justify their design decisions, especially when there are conflicting values (for instance, national security and personal privacy).

Insights from the field of accessible computing can also be useful in making security and privacy designs inclusive to a wide range of user populations. Accessible computing focuses on building technologies to improve the independence, access, and quality of life for people with disabilities. Wobbrock and colleagues propose ability-based design, which shifts the view from focusing on people's disabilities to their abilities.[15] They offer seven ability-based design principles based on their extensive experiences in designing technologies for people with disabilities. These principles include ability, accountability, adaptation, transparency, performance, context, and commodity.[15] For instance, the principle of ability states that "Designers will focus on ability not dis-ability, striving to leverage all that users can do."[15] The principle of accountability means that designers should change the systems rather than the users if the systems do not perform well.[15] These principles have proven valuable for designing accessible technologies for people with disabilities and should be adopted for inclusive S&P designs that support a wide range of under-served user groups.

Research Agenda

This preliminary research agenda of inclusive security and privacy focuses on privacy for people with visual impairments as a concrete example domain. Similar research topics could be conducted for the security and privacy needs of other under-served populations as well as the intersectionality of these populations.

Inclusive Security/Privacy Analysis

The extant literature on people's security and privacy concerns, preferences, and practices tends to focus on interviews or surveys that might not capture people's everyday experiences as they enact their privacy and security. As suggested by the literature on intersectionality, a particularly valuable addition is to study people's privacy and security experiences in their daily lives more naturally and longitudinally, for instance, using participant observation ("shadowing") and diary studies. Longitudinal diary study is a good method to understand people's mundane everyday experiences that they might forget to provide in an interview or a survey.

The extant literature on people's security and privacy concerns, preferences, and practices tends to focus on interviews or surveys that might not capture people's everyday experiences.

While the diary study approach can provide many insights into the everyday privacy challenges and practices of people with visual impairments, it has an important limitation—it's based on self-reported data. For instance, studies have shown that people with visual impairments face challenges in recognizing emergent security/privacy threats (for instance, shoulder surfing). Therefore, they may miss reporting privacy-invading incidents that they did not recognize. To address this methodological limitation, one could also conduct lightweight ethnographic studies to directly observe how people enact their security and privacy in their daily life but also help identify potential risks that the participants did not recognize. A researcher will "shadow" a participant for an extended period of time (for instance, a few days) in the participant's home and/or workplace upon permission.

In addition, critical and participatory approaches that center on people who are under-served, collect their personal stories, and conduct meta-analyses of studies would be very valuable in understanding these people's security/privacy needs and practices.

Inclusive Design and Evaluation

The prior literature and the results of the inclusive security and privacy analysis can be fed into the design of inclusive security and privacy mechanisms.

One promising design approach in this context is participatory design[16] where the design team directly includes members of the target user population (for example, children) who will actively engage throughout the design process. These participatory design sessions should engage a wide range of stakeholders including people from different under-served groups. These design sessions can be structured to explore everyone's own security and privacy concerns and practices, co-design, and pilot-test low-fidelity designs. It is important to note that the outcomes of participatory design often require designers or researchers to synthesize, select, adapt, implement, and evaluate in an iterative fashion. Once system prototypes are built, lab or field experiments can be conducted to evaluate the functionality, usability, and the broader user experience of these prototypes.

Inclusive Design Guidance Development

The goal of this research direction is to develop design guidelines for creating security and privacy designs that are inclusive to different user abilities, identities, and values. This research direction can include several components. First, inclusive security and privacy prototypes can be evaluated by existing design guidelines for privacy (such as in "Privacy as Contextual Integrity"[17]) and for accessibility and inclusion (such as in "Ability-Based Design: Concept, Principles and Examples"[15]). Second, it can include other under-served populations. Given that people from different under-served groups can differ drastically, tools designed for one under-served population may or may not be directly applicable to other under-served populations. In fact, different under-served populations may need to be studied separately, and inclusive design principles may be derived inductively from studying and designing for several specific populations. Third, research can seek to provide further design guidance for supporting other under-served populations based on evaluation results of inclusive security and privacy prototypes.

While it is desirable to derive inclusive security/privacy design patterns (that is, what/how to do) and anti-patterns (that is, what/how to avoid) that can be applied universally, practically this might be extremely difficult if not impossible due to the seemingly uncountable human characteristics. Partial rather than universal perspective is also valuable even though it can only be generalized to a limited number of under-served populations.

Making Security and Privacy Tools More Inclusive

There are several ways in which current security and privacy tools or research could be extended to make them more inclusive. For instance, user studies of security and privacy should include more under-served populations. Similarly, security and privacy risk assessments should explicitly consider under-served populations (for instance, an assessment of a social media platform should consider youth and older adults as its users). The design and evaluation of S&P technologies, especially those that involve human efforts, should include different under-served populations.

Inclusive S&P Community Building

Community building is an important aspect of supporting this new wave of research. There is an emerging community of researchers and practitioners interested in inclusive S&P. Several colleagues and I have co-organized a series of workshops on inclusive privacy and security (WIPS) at the SOUPS conferences. We discussed a wide range of user groups (such as children, older adults, people with disabilities, crime victims, and people who have little education or low socioeconomic status) and application domains (such as authentication, CAPTCHA, banking/shopping, browser security, and wearables). We also created various scenarios and conducted group design activities for these scenarios. One observation is that we still do not have a systematic methodology to support inclusive design. As discussed earlier, this is a crucial component for future research and development. In addition, a website has been launched to support this emerging research community including a curated bibliography on this topic: *www.inclusiveprivacy.org*.

The current mainstream research in security and privacy tends to focus on technical mechanisms and usability. In this article, I highlight that while these two perspectives are invaluable, they fall short of paying enough attention to other equally important issues such as accessibility and needs of many under-served user populations. The idea behind inclusive security and privacy elevates the important consideration of people's abilities, characteristics, needs, and values as first-class design requirements for security and privacy mechanisms. I encourage security and privacy researchers and practitioners to think about whether their designs can support or empower various under-served populations to protect their security and privacy. This article supports inclusive security and privacy as a promising new wave of research that both challenges and complements the dominate foci on making security and privacy mechanisms technically sound and usable. ■

Acknowledgments

This article builds upon a paper previously published in the New Security Paradigms Workshop (NSPW 17). I thank Heather Lipford, Jessica Staddon, Christine Anthony, and Charlotte Price for their help and feedback on this article. I am also grateful to many colleagues such as Anil Somayaji, Olgierd Pieczul, Apu Kapadia, Alfred Kobsa, Dan Cosley, Mike Just, Larry Koved, Karyn Moffatt, Lynne Coventry, Jeffrey Bigham, Amy Hurst, Aaron Steinfeld, John Zimmerman, Lorrie Cranor, Lujo Bauer, Jason Hong, Blase Ur, Norman Su, Jordan Hayes, Natã Barbosa, Yaxing Yao, Yun Huang as well as the NSPW 17 attendees and many colleagues at Syracuse University for discussions that helped shape my ideas. This work was supported in part by the National Institute on Disability, Independent Living, and Rehabilitation Research (NIDILRR Grant 90DP0061-01-00) and the National Science Foundation (NSF Grant CNS-1652497).

References

1. B. Dosono, J. Hayes, and Y. Wang, "I'm Stuck: A Contextual Inquiry of People with Visual Impairments

in Authentication," *Symposium on Usable Privacy and Security* (SOUPS), 2015.
2. J.H. Saltzer and M.D. Schroeder, "The Protection of Information in Computer Systems," *Proceedings of IEEE*, vol. 9, 1975, pp. 1278–1308; www.cs.virginia.edu/~evans/cs551/saltzer.
3. S. Garfinkel and H.R. Lipford. *Usable Security: History, Themes, and Challenges*, Morgan & Claypool Publishers, 2014.
4. A. Whitten and D. Tygar. "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," *Ninth USENIX Security Symposium*, 1999.
5. J. Henrich, S.J. Heine, and A. Norenzayan, "The Weirdest People in the World?," *The Behavioral and Brain Sciences,* vol. 33, nos. 2–3, 2010, pp. 61–83; doi.org/10.1017/S0140525X0999152X.
6. N. Barbosa, J. Hayes, and Y. Wang, "UniPass: Design and Evaluation of a Smart Device-Based Password Manager for Visually Impaired Users," *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing* (UbiComp 16), 2016.
7. K.E. Caine et al., "DigiSwitch: A Device to Allow Older Adults to Monitor and Direct the Collection and Transmission of Health Information Collected at Home," *Journal of Medical Systems*, vol. 35, no. 5, 2011, pp. 1181–1195; doi.org/10.1007/ s10916-011-9722-1.
8. R.L. Mace, G.J. Hardie, and J.P. Place, "Accessible Environments: Toward Universal Design," Center for Accessible Housing, North Carolina State University, 1990.
9. Y. Wang. "The Third Wave?: Inclusive Privacy and Security," *Proc. of the 2017 New Security Paradigms Workshop* (NSPW 17), 2017, pp. 122–130; doi.org/10.1145/3171533.3171538.
10. T. Ahmed et al., "Addressing Physical Safety, Security, and Privacy for People with Visual Impairments," SOUPS, 2016; https://www.usenix.org/conference/soups2016/technical-sessions/ presentation/ahmed.
11. P. Briggs and L. Thomas, "An Inclusive, Value Sensitive Design Perspective on Future Identity Technologies," *ACM Trans. Comput.-Hum. Interact.*, vol. 22, no. 5, 2015, pp. 23:1–23:28; doi.org/10.1145/2778972.
12. K. Crenshaw, "Mapping the Margins: Intersectionality, Identity Politics, and Violence against Women of Color," *Stanford Law Review*, vol. 43, no. 6, 1991, pp. 1241–1299; doi.org/10.2307/1229039.
13. D. Haraway, "Situated Knowledges: The Science Question in Feminism and the Privilege of Partial Perspective," *Feminist Studies*, vol. 14, no. 3, 1988, pp. 575–599; doi.org/10.2307/3178066.
14. B. Friedman, "Value-sensitive Design," *Interactions,* vol. 3, no. 6, 1996, pp. 16–23; https://doi.org/10.1145/242485.242493.
15. J.O. Wobbrock et al., "Ability-Based Design: Concept, Principles and Examples," *ACM Trans. Access. Comput.*, vol. 3, no. 3, 2011, pp. 9:1–9:27; doi.org/10.1145/ 1952383.1952384.
16. D. Schuler and A. Namioka, *Participatory Design: Principles and Practices*, CRC Press, 1993.
17. H. Nissenbaum, "Privacy as Contextual Integrity," *Washington Law Review Association*, vol. 79, 2004, pp. 119–158; papers.ssrn.com/sol3/papers.cfm? abstract_id=534622.

Yang Wang is with Syracuse University, SALT Lab, School of Information Studies, Syracuse, New York. Contact at ywang@syr.edu.

Editors: Melissa Dark, dark@purdue.edu | Jelena Mirkovic, mirkovic@isi.edu | Bill Newhouse, newhouse@nist.gov

Peer Instruction Teaching Methodology for Cybersecurity Education

Irfan Ahmed and Vassil Roussev | University of New Orleans

Effective cybersecurity professionals continuously adapt to the fast-changing security landscape; this requires deep analytical skills and an agile mindset. In contrast, it has been documented that the traditional lecture approach does not sufficiently engage the students in class and largely fails to stimulate the necessary thinking processes that would enable learners to quickly understand dynamically shifting attack vectors and to devise effective countermeasures in real time. In other words, students come out of security classes nominally knowing the material, yet are largely unprepared mentally for the daily rigors of the high-stakes competition between attackers and defenders.

In our experience, *peer instruction*, a well-defined teaching protocol originally developed by Eric Mazur at Harvard University,[1] holds a significant promise to deliver better cybersecurity education effectively. Peer instruction has been widely used in several disciplines such as philosophy, psychology, geology, and biology, and showed promising results on student learning. Recently, it has been explored for core computer science courses, such as Theory of Computation and Computer Architecture, and found effective in improving students' grades by 6 percent, reducing failure rates by 61 percent, and retaining students in a computer science major by 31 percent.[2–4] The students not only value peer instruction but also recommend that more instructors use it at both small colleges and large schools.

This article presents the application of peer instruction in a cybersecurity curriculum. We have developed the peer instruction material for three cybersecurity courses and evaluated them in a small group setting for a pilot study. The evaluation results show significant promise for the peer instruction in cybersecurity. The material is available at https://github.com/ahmirf/peer-instruction-questions-for-cybersecurity.

The Peer Instruction Method

In a peer instruction class, the instructor divides the original lecture into a series of questions; each question focuses on a target concept and follows the same format. The one critical prerequisite for the success of the method is a pre-class reading assignment; that is, students are required to read up on the topic covered in the class *ahead* of time. Thus, the class can be assumed to have some basic knowledge on the topic, enough to comprehend the questions during the lecture and discuss the answers with other students. The actual per-topic instruction process follows four major points:

- The instructor poses a multiple-choice question to students and gives them two to three minutes to respond individually. *Clickers* are commonly used to facilitate the logistics. They are handheld transmitters that send radio frequency signals to a receiver attached to the instructor's computer. Clickers allow students to easily respond to the question; the classroom response system automatically collects and summarizes the results for the instructor.
- If a non-trivial fraction of the responses is incorrect, the instructor asks the students to discuss their answers with a group of students sitting close by. Research shows that the real learning occurs during the discussions in this step.[5]
- In about three to four minutes, the instructor stops the discussion and polls for the answers again.
- The instructor presents the correct answer to the students and, based on the poll results, may further discuss it with them.

Developing Peer Instruction Questions for Cybersecurity

Our team studied the application of peer instruction in cybersecurity[6,7] and developed 280 peer instruction questions for three courses—Introduction to Computer Security, Network Penetration Testing, and Digital Forensics, offering an introduction to *security concepts*, a *defensive view* of cybersecurity, and an *offensive view* of cybersecurity, respectively. The first is closest to the traditional lecture format, whereas the second and third are intensive hands-on classes; together, they

Copublished by the IEEE Computer and Reliability Societies

form the basis of a broad skillset in security.

Based on our experience, we identified a basic methodology for developing peer instruction questions systematically. It consists of four steps: concept identification, concept trigger, question presentation, and question development.

Concept Identification

The first step to create a question is to identify a target concept. The student discussion during a peer instruction process and multiple choices in the question should help the students understand the concept.

Concept Trigger

After identifying a target concept, we introduce one or more concept triggers to provoke a student's thinking process and to set the desired direction for the student discussion. For instance, we deliberately introduce some ambiguity in answer choices and expect the students to identify it, bringing it to the discussion with other students. Table 1 presents a brief list of concept triggers originally proposed by Beatty and colleagues.[8]

Table 1. List of common concept triggers.*

Concept triggers	Brief description
Compare and contrast	Compare and draw conclusions from multiple situations
Identify a set or subset	Identify a subset that fulfills some criteria
Omit necessary information	Insufficient information to answer a question
Use "none of the above"	A choice to reject other options
Trap unjustified assumptions	Facilitate the identification of potential unjustified assumptions by the students
Deliberate ambiguity	Ambiguous answer choices for a question
Trolling for misconceptions	Answer choices have common misconceptions by the students
Oops-go-back	Pair of questions; first traps the students with a common error; the second clarifies it
Require unstated assumptions	A question misses required assumption, potentially leading to multiple defensible answers

** A comprehensive list is presented in Beatty and colleagues'[8] and Johnson and colleagues' work.[6,7]*

Question Presentation and Development

After identifying the potential concept triggers for a question, we determine how the question should be presented to students to facilitate understanding. A question can be presented in different ways including scenario, example, or diagram. The question is then articulated consisting of the text and multiple choices along with any other supporting content such as a diagram, graph, and so on.

Example Peer Instruction Questions

The following examples of peer instruction questions cover a cyberattack, reverse engineering, and digital forensics; each example also presents a brief analysis of the questions in terms of concept trigger and each question's presentation.

Q1: Cyberattack

Which of the following represents an example of an attack?

a) A web application is vulnerable to cross-site scripting.
b) A successful denial-of-service attack takes down a login page for a full day.
c) A database administrator implements a system to encrypt data stored in the database.
d) A SQL injection vulnerability is used to obtain sensitive information by an unauthorized user.
e) More than one of these.

Analysis of Q1: This question uses the concept trigger "identify a set or subset," because both B and D are examples of an attack, and thus, the correct answer is E. The question is presented as an example, and an important detail here is that the examples are in the choices rather than the question description.

Q2: Reverse Engineering

After executing the instructions in Figure 1 while single-stepping inside a debugger on an 80486 processor, what is the value of the 16-bit word at location `loc_10129+5`?

a) 168h
b) 152h
c) 4D4Ch
d) Value is unknown
e) None of the above

Analysis of Q2: The question uses the concept trigger "analysis and reasoning" to allow students to discuss the answers based on whether instruction prefetch caching is enabled. It also provides "none of the above" to allow students to discard other choices. If prefetch caching is enabled, the correct answer is 4D4Ch.

```
1.    Start:
2.    mov word ptr loc_10106 + 1, 152 h
3.    loc_10106: ;DATA XREF
4.    mov ax, 168h
5.    mov word ptr loc_10129 + 5, ax
6.    loc_10129:  ;DATA
7.    mov word ptr es: 0, 4D4Ch
```

Figure 1. Self-modifying code snippet.

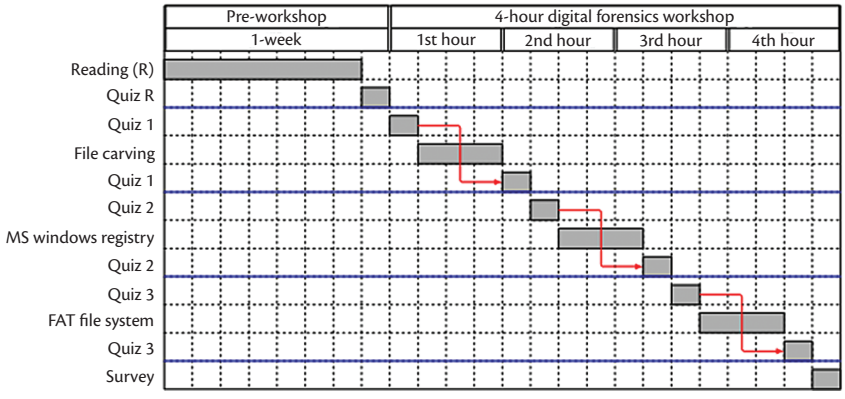


Figure 2. Timeline of the workshop activities including quizzes, survey, and peer instruction lectures on three digital forensics topics, file carving, MS Windows registry, and FAT filesystem.

Q3: Digital Forensics

In which of the following situations is file carving most effective?

a) The targeted drive is highly fragmented.
b) The targeted drive has been defragmented before carving.
c) The system being used to examine the drive has low free disk space.
d) The system being used to examine the drive has high free disk space.
e) More than one of the above

Analysis of Q3: This question has a choice "More than one of the above" to utilize the concept trigger "identify a set or subset." It also uses "trolling for misconceptions" because the defragmentation rearranges data in contiguous disk blocks; however, it may lose some deleted files. The correct answer is D because high free disk space leads to less fragmentation, which helps file carving recover more data.

Peer Instruction Evaluation

As already mentioned, we developed 280 questions for three cybersecurity courses—Introduction to Computer Security, Digital Forensics, and Network Penetration Testing. We used a small subset of the forensics questions in a four-hour workshop on digital forensics to perform a pilot study on the effectiveness of peer instruction for a cybersecurity course. The workshop covered three topics—file carving, MS Windows registry, and FAT filesystem. Figure 2 presents the timeline of the workshop activities. Twelve undergraduate students attended the workshop, consisting of 11 males and one female, who had never taken any prior computer forensics course. We provided reading material to the students on the topics a week before the workshop. We also distributed a quiz of five questions on the material using Google Forms and asked the students to complete the quiz before the workshop. We used the quiz to ensure that the students read the material.

During the workshop, the students were divided into four groups for peer discussion. The entire workshop was centered on seven peer instruction questions: two for each topic and one for the introductory discussion on computer forensics. The clickers were used to collect student responses on the questions.

For the evaluation, we used the following metrics: quiz, survey, and clicker responses. The quizzes were presented twice to the students, once before and once after a topic, to quantify student learning. The survey gathered the data on prior usage of clickers, workshop preparation (reading material and quiz), peer discussion, clicker usage, and lecture pacing. We employed a survey developed by Beth Simon and Leo Porter of the University of California, San Diego, and Cynthia Lee of Stanford University that had been used in numerous peer instruction courses.[2,9,10]

In the evaluation results, the quizzes (at the beginning and end of the topics) and peer instruction questions (before and after discussions) showed clear evidence of student learning gain. Table 2 presents the results of the survey.[6] The results were similar to the previous studies of peer instruction in the core computer science courses.[1–4] For instance, 92 percent of students reported that discussions with peers during a lecture helped them understand the workshop material. Ninety-one percent of them not only found the immediate feedback from clickers very useful but also

recommended that other instructors use peer instruction in their courses.

The study showed that the students quickly adapted to the format, found it useful, and highly recommended the approach be extended to a wider range of subjects. To further substantiate our results, we plan to perform a large-scale study of peer instruction on cybersecurity courses.

Although we have a positive experience with peer instruction, we notice that a big challenge for an instructor to adopt this technique is to manage time and maintain an appropriate pace for a class lecture. We encourage other educators to employ peer instruction as an active learning technique in their courses and share their experiences with the cybersecurity education community. ■

Table 2. Results of the peer instruction survey on the digital forensics workshop.

Questions	Average opinion
Discussing course topics with my seatmates helped me better understand the material.	92%
The immediate feedback from clickers helped me focus on weaknesses in my understanding of the workshop material.	91%
I recommend that other instructors use this approach (reading quizzes, clickers, in-class discussion) in their courses.	91%
I read the required material before the workshop.	89%
Clickers are an easy-to-use class collaboration tool.	89%
Thinking about clicker questions on my own, before discussing with people around me, helped me learn the workshop material.	87%
Most of the time my group actually discussed the clicker question.	87%
Clickers helped me pay attention in this workshop compared to traditional lectures.	82%
Generally, by the time we finished with a question and discussion, I felt pretty clear about it.	80%
Using clickers with discussion is valuable for my learning.	80%
The pre-workshop reading quiz helped me recognize what was difficult in the reading.	78%
Knowing the right answer is the only important part of the clicker question.	46%

Acknowledgments

This work was supported by the NSF grant 1500101.

References

1. C.H. Crouch and E. Mazur, "Peer Instruction: Ten Years of Experience and Results," *American Journal of Physics*, vol. 69, no. 9, 2001, pp. 970–977.
2. L. Porter et al., "Peer Instruction in Computer Science at Small Liberal Arts Colleges," *Proceedings of the 18th Annual Conference on Innovation and Technology in Computer Science Education*, July 2013.
3. B. Simon et al., "Experience Report: Peer Instruction in Introductory Computing," *Proceedings of the 41st SIGCSE Technical Symposium on Computer Science Education*, March 2010.
4. B. Simon, J. Parris, and J. Spacco, "How We Teach Impacts Student Learning: Peer Instruction vs. Lecture in CS0," *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, March 2013.
5. L. Porter et al., "Peer Instruction: Do Students Really Learn from Peer Discussion," *Proceedings of the 7th Annual International Computing Education Research Workshop*, August 2011.
6. W. Johnson et al., "Peer Instruction for Digital Forensics," *USENIX Advances in Security Education Workshop* (ASE 17), August 2017.
7. W. Johnson et al., "Development of Peer Instruction Questions for Cybersecurity Education," *USENIX Workshop on Advances in Security Education* (ASE 16), 2016.
8. L. Beatty, W. Gerace, and R. Dufresne, "Designing Effective Questions for Classroom Response System Teaching," *American Association of Physics Teachers*, vol. 74, no. 1, 2006.
9. C.B. Lee, S. Garcia, and L. Porter, "Can Peer Instruction Be Effective in Upper-Division Computer Science Courses?," *Trans. Comput. Educ.*, vol. 13, no. 3, 2013, pp. 12:1–12:22.
10. L. Porter et al., "A Multi-Institutional Study of Peer Instruction in Introductory Computing," *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (SIGCSE 16), 2016, pp. 358–363.

Irfan Ahmed is in the Department of Computer Science at the University of New Orleans. Contact at irfan@cs.uno.edu.

Vassil Roussev is in the Department of Computer Science at the University of New Orleans. Contact at vassil@cs.uno.edu.

Editor: Pavel Gladyshev, pavel.gladyshev@ucd.ie

Cryptocurrencies—A Forensic Challenge or Opportunity for Law Enforcement?

An INTERPOL Perspective

Giannis Tziakouris | INTERPOL

The anonymous and decentralized nature of cryptocurrencies has turned them into a powerful weapon in the cyber-arsenal of national and international criminal groups by facilitating their illicit activities while evading prosecution. This has baffled the international law enforcement community, with a large number of cryptocurrency-related investigation cases received on a monthly basis by INTERPOL.

Cryptocurrencies and Crime

Cryptocurrencies have been used extensively in darknet markets for receiving payments for illicit services such as distributed denial of service; malware binaries; botnets; and the purchase of illegal products including weapons, drugs, and falsified or stolen documents. In particular, darknet markets such as Silkroad, AlphaBay, and Hansa were reaping large profits, with the last reaching $3,000,000 between September 2015 and December 2016.[1]

Furthermore, it is important to note their use by extremist groups to facilitate the trade of illicit products (for instance, stolen antiquities, drugs, and firearms), remit money to areas that are under high financial scrutiny or embargo, and publicly crowd-fund their operations.

Money laundering through cryptocurrencies is another major challenge for law enforcement. A significant number of criminals have established cryptocurrency exchanges or initial coin offering (ICOs) with the goal of laundering illicit profits. A few well-known money laundering cases include the Bitcoin exchange OKCoin with hundreds of thousands of US dollars laundered[2] as well as the case of BitInstant, in which an estimated sum of more than $1,000,000 was laundered for Silk Road market customers.[3]

Moreover, cryptocurrencies have advanced the operations of various malware families such as ransomware, with CryptoLocker and CryptoWall receiving 133,045.9961 BTC and 87,897.8510 BTC, respectively;[4] cryptojacking, with Jenkins-Miner earning its operator over $3,000,000 worth of Monero;[5] and crypto-stealing Trojans, such as CryptoShuffler, which stole hundreds of thousands of US dollars by targeting the contents of volatile memory—that is, the clipboard.[6]

Another method through which criminals exploit the blockchain is by injecting arbitrary encoded data chunks (for instance, pictures) in non-standard Bitcoin transactions to disseminate child exploitation material.[7] The biggest challenge for law enforcement agencies here is

Copublished by the IEEE Computer and Reliability Societies

the immutable nature of the blockchain that disallows the removal of embedded illicit content.

Furthermore, there has been a significant rise in ICO exit scams where criminals persuade their victims to buy large numbers of fake coins, subsequently disappearing with millions of dollars.

Lastly, cryptocurrencies are used for sponsoring nation-state attacks, as a number of countries around the world are highly affected by the existence of contemporary hybrid-war strategies.

Adapting Law Enforcement and Criminal Strategies

As cryptocurrencies are associated with a plethora of crime types such as narcotics, firearms, money laundering, terrorism, and child exploitation, the international law enforcement community, including INTERPOL, has begun to focus on mastering the blockchain. In doing so, a significant amount of resources has been allocated for the exploration of the use of cryptocurrencies by criminals as well as the development of proprietary analytical tools for tracing cryptocurrency transactions.

In policing, two different schools of thought exist, with the first perceiving cryptocurrencies as a threat and the second viewing them as an investigational opportunity. The first group considers cryptocurrencies a disruptive solution enabling criminals to facilitate their illegal activities in the absence of policing, hence calling for its prohibition. On the other hand, a small yet growing law enforcement community views cryptocurrencies as an investigation opportunity where criminally associated information is now publicly and permanently indexed in the blockchain to be analyzed for the extraction of valuable forensic data that can lead to attribution and prosecution.

In recent years, there has been significant effort from both the industry and various law enforcement agencies, including INTERPOL, to develop forensic tools and methodologies for the analysis of various cryptocurrencies, with the majority focusing on the Bitcoin network. The vast focus on the analysis of Bitcoin transactions can be attributed to the substantial number of criminal cases affiliated with it, despite the existence of more anonymous cryptocurrencies. The increase in the value of Bitcoin and its wide adoption by markets (which offers easy entry and exit points) have played a catalytic role in boosting the magnitude of the criminal cases and trends associated to it. Despite the wide adoption of bitcoins by criminals, the recent success stories of contemporary analytical tools that allowed police investigators to partially de-anonymize the Bitcoin network and reveal the identity of criminals have caused a shift in the use of cryptocurrencies. An increasing number of criminals are using Bitcoin only as an entry and exit point; in the interim, they trade their bitcoins for more anonymous cryptocurrencies, in particular, Dash, Monero, Zcash, PIVX, Verge, and Namecoin.

It is imperative for law enforcement agencies to co-evolve with the current state of the art and identify and thwart online criminal activities that are linked to cryptocurrencies.

Law enforcement considers the aforementioned cryptocurrencies highly disruptive due to their enhanced anonymity, which makes them an effective weapon for criminals. Dash and Zcash enable users to keep their activity history and balances private, which ultimately restricts law enforcement investigators from identifying and tracing suspicious transactions. Similarly, Monero uses ring signatures, ring confidential transactions, and stealth addresses to obfuscate the origins, amounts, and destinations of transactions. Furthermore, PIVX implements the Zerocoin protocol that converts public PIV into anonymous PIV (aka zPIV) to conceal the pseudo-identity of the sender or any traces that can lead to the real identity of the sender. Verge is another anonymous cryptocurrency that leverages the wraith protocol to enable its users to switch between public and private ledgers. When the wraith protocol is turned on, the transaction data is hidden. Finally, Namecoin does not share the same strong anonymity characteristics or goals as the cryptocurrencies above but is still identified as a potential threat to policing due to its feature that allows criminals to anonymously register illegal websites without providing any personal information, thus preventing investigators from identifying the administrators behind these pages.

As an extra layer of protection (despite the enhanced level of privacy offered by the aforementioned cryptocurrencies), many criminals use crypto-mixing/tumbling services or decentralized P2P exchange markets to "clean" their "tainted" coins, making it increasingly difficult for police investigators to follow their transactions.

To counteract the illicit use of cryptocurrencies, law enforcement is now focusing on the development of advanced solutions for tracing criminally linked transactions. In particular, police agencies work toward developing forensic tools for the analysis of various computing

devices for the identification of cryptocurrency-related artefacts, such as wallets, and cryptocurrency hashes; fingerprinting tools for identifying the use of mixers/tumblers in cryptocurrency transactions; clustering solutions for the aggregation of the addresses belonging to the same criminal actors for better attribution; and cross-ledger tracing tools to support the association of suspicious transactions in different blockchains.

In addition to the current work by law enforcement on a national level, INTERPOL acts as an information hub on the international level by bringing police investigators from various nations, researchers, and blockchain developers together to share best investigation practices and forensic tools for cryptocurrencies. Moreover, INTERPOL works toward developing the investigation capabilities of its member countries by delivering advanced hands-on trainings on cryptocurrencies. INTERPOL strives for innovative solutions by seeking partnerships with the public and private sectors, including cybersecurity and cryptoanalytic companies. INTERPOL held its first international Darknet and Cryptocurrencies Working Group in March 2018 to further stimulate discussions surrounding policing solutions for cryptocurrency-related crimes, identifying Altcoins and cross-ledger investigations as the biggest challenges for law enforcement.

Despite the numerous challenges that the international law enforcement community faces when investigating cryptocurrencies, a number of investigation opportunities do exist. The blockchain is here to stay due to the wide use of cryptocurrencies by investors, adopters, and pioneers. While it is anticipated that a large number of its characteristics will significantly adapt in the near future to overcome critical technological shortcomings such as its size and scalability, its use for illicit activities will only continue to grow. It is imperative for law enforcement agencies to co-evolve with the current state of the art and identify and thwart online criminal activities that are linked to cryptocurrencies. For better efficacy in combatting cryptocurrency-related crime, an implementation of an international understanding and a legal framework to regulate it should be considered; this will enable law enforcement to access information for criminally linked transactions and urge cryptomarkets and exchanges to enforce strong KYC (know your client) policies. However, while its purpose is to apprehend criminals, actions taken by policing should not tamper with the confidentiality, integrity, and availability attributes of the blockchain and the development of other innovative solutions. ■

References

1. "OnionScan Report: Reconstructing the Finances of Darknet Markets through Reputation Systems," Mascherari Press, 2018; https://mascherari.press/onionscan-report-forensic-finances-dark-markets.
2. "Bitcoin Exchange OKCoin Fined in Money Laundering Case," Gautham, 15 Aug. 2016; https://www.newsbtc.com/2016/08/15/china-okcoin-exchange-fined.
3. J. Pagliery, "Bitcoin Exchange CEO Arrested for Money Laundering," CNN Tech, 28 Jan. 2014; http://money.cnn.com/2014/01/27/technology/security/bitcoin-arrest.
4. M. Conti, A. Gangwal, and S. Ruj, "On the Economic Significance of Ransomware Campaigns: A Bitcoin Transactions Perspective," arXiv: 1804.01341v4 [cs.CR], 27 Apr. 2018.
5. M. Osena, "Cryptocurrency-Mining Malware: 2018's New Menace?," Trend Micro blog, 28 Feb. 2018; https://blog.trendmicro.com/trendlabs-security-intelligence/cryptocurrency-mining-malware-2018-new-menace.
6. "CryptoShuffler: Trojan Stole $140,000 in Bitcoin," Kaspersky Lab Daily, 31 Oct. 2017; https://www.kaspersky.com/blog/cryptoshuffler-bitcoin-stealer/19976.
7. R. Matzutt et al., "A Quantitative Analysis of the Impact of Arbitrary Blockchain Content on Bitcoin," RWTH Aachen University, Germany.

Giannis Tziakouris is a digital crime analyst at INTERPOL. Contact at g.tziakouris@INTERPOL.INT.

IEEE computer society

PURPOSE: The IEEE Computer Society is the world's largest association of computing professionals and is the leading provider of technical information in the field.
MEMBERSHIP: Members receive the monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.
COMPUTER SOCIETY WEBSITE: www.computer.org
OMBUDSMAN: Direct unresolved complaints to ombudsman@computer.org.
CHAPTERS: Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.
AVAILABLE INFORMATION: To check membership status, report an address change, or obtain more information on any of the following, email Customer Service at help@computer.org or call +1 714 821 8380 (international) or our toll-free number, +1 800 272 6657 (US):

- Membership applications
- Publications catalog
- Draft standards and order forms
- Technical committee list
- Technical committee application
- Chapter start-up procedures
- Student scholarship information
- Volunteer leaders/staff directory
- IEEE senior member grade application (requires 10 years practice and significant performance in five of those 10)

PUBLICATIONS AND ACTIVITIES
***Computer*:** The flagship publication of the IEEE Computer Society, *Computer*, publishes peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.
Periodicals: The society publishes 13 magazines, 19 transactions, and one letters. Refer to membership application or request information as noted above.
Conference Proceedings & Books: Conference Publishing Services publishes more than 275 titles every year.
Standards Working Groups: More than 150 groups produce IEEE standards used throughout the world.
Technical Committees: TCs provide professional interaction in more than 30 technical areas and directly influence computer engineering conferences and publications.
Conferences/Education: The society holds about 200 conferences each year and sponsors many educational activities, including computing science accreditation.
Certifications: The society offers two software developer credentials. For more information, visit www.computer.org/certification.

COMPUTER SOCIETY OFFICES

Washington, D.C.: 2001 L St., Ste. 700, Washington, D.C. 20036-4928
Phone: +1 202 371 0101 • **Fax:** +1 202 728 9614
Email: hq.ofc@computer.org
Los Alamitos: 10662 Los Vaqueros Circle, Los Alamitos, CA 90720 **Phone:** +1 714 821 8380
Email: help@computer.org

MEMBERSHIP & PUBLICATION ORDERS
Phone: +1 800 272 6657 • **Fax:** +1 714 821 4641 • Email: help@computer.org
Asia/Pacific: Watanabe Building, 1-4-2 Minami-Aoyama, Minato-ku, Tokyo 107-0062, Japan
Phone: +81 3 3408 3118 • **Fax:** +81 3 3408 3553
Email: tokyo.ofc@computer.org

revised 03 July 2018

Daniel E. Geer Jr.
In-Q-Tel

You Are What You Eat

The Internet is an omnivore; it will take anything that anything that can (talk to anything that can) send a packet sends. Every line of application glue code there is exists for one reason—to make omnivorous ingestion possible. Every data analytic program scoring itself on how well it handles unstructured inputs is doing so to satisfy an omnivore's dream. David Wheeler's bon mot, "All problems in computer science can be solved by another level of indirection," is our version of the Omnivore's Dilemma.

A non-negligible fraction of Internet backbone traffic cannot be identified as to protocol—yet something eats it. Who? Qualcomm's Swarm Lab at UC Berkeley predicts 1,000 radios per human by 2025. What will consume that stream? Pete Diamandis' book *Abundance* calls for 45 trillion networked sensors by 2034. That's a couple of orders of magnitude more than the number of neurons in the human body; what brain can or will consume that flow? The Internet of Things' installed base is enjoying something like a 35 percent compound annual growth rate. The number of hosts with an advertised DNS record is growing much more slowly, that is, the 35 percent compound annual growth rate is a nameless multitude; think krill. Something eats those outputs.

The number of transistors on the larger chips is now 20 billion and up (Xilinx Everest, AMD Epyc). A hidden kill switch in hardware might take 5,000 of them; that's 250 parts per billion. The US limits (naturally occurring) ethyl carbamate in wine to 15 parts per billion.

The size of code blobs submitted to static analysis companies had been growing into the tens of gigabytes (meaning code written by machines). Now, sub-50MB code blobs for microservices are beginning to dominate. Yet we all know that security is not composable just like we know to expect drug interactions between even a few prescriptions.

The number of truly competent cybersecurity people is far less than the demand. The Big Five are out to hire (eat) them all regardless of the cost. Some 3.5 million cybersecurity jobs will be unfilled by the next US Inauguration Day. Let them eat cake.

Likewise, the number of truly relevant cybersecurity ideas is far less than the demand. De novo cybersecurity investment has grown at 25 percent compound annual growth rate for 15 years now; Q42017 saw 279 cybersecurity startups receive more than $800 million in venture capital.

States, like kings of old, don't trust what comes up from the kitchen. (Obama had, and Putin has, food tasters.) The Chinese government wants Chinese networking gear in China. The US government wants US networking gear in the US. The Russian government wants Russian networking gear in Russia. Smaller countries just get to choose whose vulns they'll share (at whose table they will eat).

The more powerful the tech, the more dual use it is. When the DARPA Cyber Grand Challenge culminated 4 August 2016, Mike Walker, its program manager, said "I cannot change the reality that all security tools are dual-use."

Food is dual use; the US population's biggest source of ill health is too much to eat. The Internet, eating everything in sight, is already too big to get out of bed. As Matthew Stewart wrote in *The Atlantic*,

> *By now we're thankfully done with the tech-sector fairy tales in which whip-smart cowboys innovate the heck out of a stodgy status quo. The reality is that five monster companies—you know the names—are worth about $3.5 trillion combined, and represent more than 40 percent of the market capital on the NASDAQ stock exchange. Much of the rest of the technology sector consists of virtual entities waiting patiently to feed themselves to these beasts.* ■

Daniel E. Geer Jr. is the chief information security officer of In-Q-Tel. Contact him at dan@geer.org.

Copublished by the IEEE Computer and Reliability Societies

IEEE computer society

SHARE THE GIFT OF KNOWLEDGE:

Give Your Favorite Student a Membership to the IEEE Computer Society!

With an **IEEE Computer Society Membership**, your student will be able to build their network, learn new skills, and access the best minds in computer science before they're even out of school. Your gift includes thousands of key resources that will quickly transition them from classroom to conference room, such as:

- **A subscription to *Computer* magazine** (12 issues per year)
- **A subscription to *ComputingEdge*** (12 issues per year)
- **Local chapter membership**
- **Full access to the Computer Society Digital Library**
- **Eligible for 3 student scholarships where we give away US$40,000 yearly**
- **Skillsoft:** Learn new skills anytime with access to 3,000 online courses, 11,000 training videos, and 6,500 technical books.
- **Books24x7:** On-demand access to 15,000 technical and business resources.
- **Unlimited access to computer.org and myCS**
- **Conference discounts**
- **Members-only webinars**
- **Deep member discounts** on programs, products, and services

Give Your Gift at: www.computer.org/2018gift

Keep up with the latest IEEE Computer Society publications and activities wherever you are.

Follow us:

- @ComputerSociety, @ComputingNow
- facebook.com/IEEEComputerSociety, facebook.com/ComputingNow
- IEEE Computer Society, Computing Now
- youtube.com/ieeecomputersociety
- instagram.com/ieee_computer_society

IEEE computer society