The error rate of algorithm analyses

Daniel J. Bernstein[1,2]

[1] Department of Computer Science, University of Illinois at Chicago, USA
[2] Institute of Information Science, Academia Sinica, Taiwan
djb@cr.yp.to

Abstract. Analyses of the accuracy and cost of algorithms sometimes turn out to be incorrect. A claimed proof can have mistakes. A claimed theorem can be false. Even when a theorem is correctly proven, an algorithm analysis can have unproven components beyond the theorem, and there can be mistakes in those components.

The literature sometimes suggests that algorithm-analysis errors are rare, but sometimes suggests that such errors are more common. The actual rate of errors is important for risk assessment and risk mitigation. A high rate would help justify investment in techniques to increase the assurance level for proofs, and for algorithm analyses beyond proofs.

This paper considers a particular strategy to determine, with high assurance, an interval containing the actual error rate of a specified pool of algorithm analyses. This paper presents reasons to believe that the interval produced by this strategy will be much smaller than the trivial interval $[0, 1]$. This paper also presents reasons to believe that this strategy is feasible to carry out.

1 Introduction

In 2021, the Australian government agreed to pay 1.8 billion Australian dollars to hundreds of thousands of victims of the government's "robodebt" scheme. The government had spent years issuing accusations of welfare fraud—often ruining lives—on the basis of a flawed algorithm. Under the applicable law, people were entitled to receive welfare payments during each two-week period of low salary; the algorithm incorrectly assumed that salary payments were spread uniformly over longer periods of time. See, e.g., [65] and [79] for short summaries of the damage, and [92] for an official post-mortem issued in 2023.

A typical response to flawed algorithms is to ask for algorithms to be *analyzed*. For example, [92, Recommendation 17.1] said that "algorithms should be made available, to enable independent expert scrutiny". O'Neil's 2016 book "Weapons of math destruction" [85] covers many further examples of decisions that had

Permanent ID of this document: 9858e116116538a8c2b53be91ff830744c868398. Date: 2025.02.03.

been delegated to flawed algorithms; [85, Conclusion] recommended "algorithmic audits".

There is a long history of literature analyzing the accuracy and cost[3] of a broad range of different algorithms. Simple examples of algorithm analysis are a core part of the standard curriculum for computer-science undegraduates. A typical textbook, "Introduction to algorithms" [52] by Cormen, Leiserson, Rivest, and Stein, says the following:

> This book will teach you techniques of algorithm design and analysis so that you can develop algorithms on your own, show that they give the correct answer, and understand their efficiency.

The word "show" refers to mathematical proofs, which—in theory; we'll return to reality later—are rigorous chains of logic guaranteeing that the algorithms always work as advertised.[4] Algorithms and analyses in the literature often tackle tasks much more sophisticated than the financial computations from [92]. Even better, beyond covering all possible inputs to an algorithm, proofs often cover many algorithms at once. This saves analysis time, and sometimes it sheds light on the question of what algorithms *cannot* do—a central question in computational complexity theory and in cryptography, where these multi-algorithm proofs are often labeled as "proofs of security".

Structurally, it is clear that an algorithm merely tested on some examples—or designed in the first place to work for those examples—can have failure cases caught by an analysis that considers what happens in all cases. However, the fact that algorithm analysis sometimes catches errors does not mean that algorithm analysis is error-free. On the contrary, an algorithm analysis can be incorrect, as the following example illustrates.

A paper [90] posted in October 2021 claimed that a specified quantum algorithm at a specified cost would solve "SVP", the famous problem of finding minimum-length nonzero vectors in lattices. Note that *testing* quantum algorithms is difficult without big quantum computers, but one can still *analyze* them and in particular prove theorems about them. Analyses of

[3] Cost is not always an issue. However, often a user is unable or unwilling to incur the costs of running an algorithm to completion; see, e.g., [96]. An analysis that ignores cost might have promised that the algorithm would deliver an accurate answer; in reality, the user has not received an answer. Even if the user finds this less damaging than a wrong answer, the user ends up dissatisfied (which is good in some cases—for example, perhaps this is an algorithm to attack cryptography). So it is unsurprising for algorithm analyses to include analyses of cost. Beware that the literature often uses the phrase "analysis of algorithms" to refer *solely* to analyses of cost, excluding analyses of accuracy: for example, [52, Section 2.2] says that "***Analyzing*** an algorithm has come to mean predicting the resources that the algorithm requires".

[4] Note that this structure inherently requires specifications of what the algorithms are supposed to do. There are some types of algorithms for which the only specifications that seem easy to write down are case-by-case tests: see, e.g., [87] on difficulties in specifying what it means for a neural network to do a good job of recognizing faces.

the capabilities of quantum algorithms motivate the ongoing upgrade to post-quantum cryptography; within post-quantum cryptography, these analyses influence decisions of which cryptographic systems to use.

I wrote to the authors a week and a half later asking a question about how [90, Corollary 7] was supposed to follow from [90, Theorem 6]. I didn't hear back. I then tweeted that I was skeptical about [90, Corollary 7], and said why. A month later, after a closer look, I tweeted that I was skeptical about [90, Theorem 6], and said why. See [22] and [23] ("aren't half of these nested amplifications?"). In September 2022, the authors retracted their paper, writing that "A reviewer pointed out an error in the amplitude amplification step in the analysis of Theorem 6".

It's easy to respond that this example simply demonstrates the importance of peer review. The paper [90] was an unrefereed preprint; the mistake was rapidly caught. The same comment applies to, e.g., a paper [49] in 2024 that claimed a faster quantum algorithm for another lattice problem; [49] was withdrawn nine days after it was posted.

However, the following example is not so easy to dismiss. This is an example where a *refereed* algorithm analysis—in fact, two such analyses—went disastrously wrong.

The topic of this example is a cryptographic construction called "XCB" that converts a standard "block cipher" AES into a disk-encryption mechanism XCB-AES. A refereed paper from 2007 [80] presented a "proof of security" supposedly showing that any algorithm breaking XCB-AES could be converted into an algorithm at similar cost breaking AES. No fast algorithm is known today to break AES, so this "proof of security" supposedly says that no fast algorithm is known to break XCB-AES. XCB was standardized in 2010.

Years later, a refereed paper [48] pointed out an exploitable flaw in the original "proof". This paper explained a fast attack breaking XCB-AES for unusual disk-block lengths. On the other hand, [48] also claimed to provide a "new security theorem" for XCB for normal disk-block lengths.

In October 2024, two independent papers, namely [102] and [39], presented fast attacks breaking XCB-AES for normal disk-block lengths. The papers do not seem to have been refereed yet, but [39, Section 4.3] explains a fatal flaw appearing in the refereed "proofs" from both [80] and [48]; [39] also reports that the authors of [80] concur. Evidently this algorithm-analysis flaw had escaped the notice of the authors *and* reviewers of [80] and [48].

A cryptography textbook by Katz and Lindell [70, page 20] claims that "Proofs of security give an iron-clad guarantee—relative to the definition and assumptions—that no attacker will succeed". The only way to reconcile this claim with the XCB example is to say that [80] and [48] were not actually providing "proofs of security". In other words, this "iron-clad guarantee" is not actually for everything *labeled* as "proofs of security", but only for the *correct* proofs—whichever ones those might be. This begs the question of how many more "proofs of security" are also wrong. More broadly, it is natural to ask how often algorithm analyses are wrong.

2 Weak evidence that algorithm-analysis errors are common

This section gives examples of how various sources can lead readers to think that the rate of algorithm-analysis errors is high. However, this section also pinpoints various reasons that the cited sources are not demonstrating a high rate of errors. See also Section 3 for various sources that can lead readers to think that the rate of algorithm-analysis errors is low.

2.1 Beware of bugs

In 1977, Knuth [55, page 7 in cited PDF] wrote the following, after presenting (pseudo)code for a particular algorithm: "Beware of bugs in the above code; I have only proved it correct, not tried it." A Google Scholar search for the quoted phrase "beware of bugs in the above code" finds 245 results as of January 2025, including 242 naming Knuth.

This call for caution can be understood as suggesting that algorithm-analysis errors are a common occurrence; but this is not even stated explicitly, let alone quantified or justified. The warning can instead be explained as Knuth's text being preliminary material: it was a classroom note, not a paper.

2.2 Proofs are rarely examined

A 1994 "security proof" from Bellare and Rogaway [10] was flawed, as pointed out by Shoup [97] several years later. In a 2007 paper "Another look at 'provable security' " (first posted as a preprint in 2004), Koblitz and Menezes [74] wrote the following:

> One has to wonder how many "proofs" of security are ever read carefully with a critical eye. The purported proof of Bellare–Rogaway in [7] was short and well-written, and the result attracted much interest (and caused OAEP to be included in the SET electronic payment standard of MasterCard and Visa [3]). If this proof went essentially unexamined for seven years, one cannot help asking whether the lengthy and often poorly-written "proofs" of less famous security claims are ever read carefully by anyone.

But asking questions is not the same as providing evidence. My papers [16, Section 7], [17], [37, Appendix A], [19, Sections 4.1 and 6], and [25, Appendix A.4] report discoveries of errors in several more "security proofs", and [75] surveys further examples,[5] but this does not address the question of how frequent such failures are.

Similarly, regarding proofs more broadly (not just "proofs of security"), Voevodsky claimed in 2014 [101] that a "technical argument by a trusted author,

[5] [75] also surveys examples of a different problem, namely theorems being treated as saying more than they actually do.

which is hard to check and looks similar to arguments known to be correct, is hardly ever checked in detail"; but the evidence provided for this claim in [101] consists of a series of anecdotes. Those stories are troubling—for example, published "theorems" were sometimes applied for years with nobody checking them—but they do not contradict the idea that such failures are rare. Similar comments apply to a 1972 claim from Davis [54] that "most proofs in research papers are unchecked other than by the author".

2.3 Corrections are suppressed

In a 2013 paper "Errors and corrections in mathematics literature" [62] (this is another source considering proofs generally, not specifically algorithm analyses), Grcar wrote the following:

> In summary, the mathematics literature observably has low correction rates, yet systemic considerations suggest that mathematicians do not have lower error rates than other researchers. This mismatch may stem from a cultural emphasis on perfection that discourages discussing mistakes. The consequences are an absurdly high expectation for peer review to catch all errors and a neglect of policies to correct the literature once published.

It is, however, easy to see gaps in the "systemic considerations" in [62]. For example, [62] claims that there is "no reason" to think that "mathematicians make mistakes less often than other researchers"; but the fact that a complete proof can in theory be recognized by pure logic is such a reason.

As another example, [62] observes that correction procedures are much less visible in editorial guidelines for mathematics than in editorial guidelines for other areas of science; [62] attributes this to a claimed pattern of mathematicians suppressing errors, rather than to proofs being more reliable than other scientific mechanisms. A few examples are given in [62] of proof errors being suppressed, but this is weak evidence for claims of a broader pattern.[6]

2.4 One reviewer found many errors

In 2022, Lamport posted a 2-page report "Some data on the frequency of errors in mathematics papers" [76]. This report says the following:

- There is "plenty of anecdotal evidence of errors in papers published in mathematical journals".
- A search found no "published data on the frequency of those errors"—beyond [62] reporting the (low) rate of *corrections*, which is not the same question.

[6] For example, one might dismiss the Mittag-Leffler incident summarized in [62] as being an isolated incident from the 19th century. As another example, there is a footnote in [62] pointing to a much more recent incident described in [66], but [66] says that the first author of the erroneous paper in question was "a professor of political science"; proofs are not a standard part of the political-science curriculum.

- Lamport had gone through all 84 public paper reviews written by one mathematician (George Bergman), and had found that 28 of those reviews reported errors ("what I thought to be serious errors, not easily corrected mistakes"), including "11 reviews reporting incorrect results" (meaning that a claimed "theorem" is false).

However, even if the numbers in [76] are correct,[7] this could simply reflect carelessness in papers in one specific area of mathematics, perhaps further biased by how papers were selected for Bergman to review. As noted in [76]: "They were all in Bergman's area of expertise. (He is an algebraist specializing in ring theory.)"

There is a counterargument in [76]: "I know of no reason to expect the number of errors in other fields of math to be significantly different." But it is easy to imagine such reasons. Consider, for example, the comment from Gowers [61] that, for *some* types of mathematics, "progress is often the result of clever combinations of a wide range of existing results". A "proof" that relies on many existing "theorems" is unsupported if even *one* of those "theorems" is wrong.

More to the point, even if errors are common in pure mathematics (which, again, goes beyond what is shown in [76]), this does not imply that errors are common in algorithm analysis. One can imagine, for example, that algorithm analyses are almost always carefully checked, exactly because of the real-world importance of algorithms.

2.5 Actions speak louder than words

There are some papers (e.g., [27]) reporting computer verification of proofs, sometimes including proofs regarding algorithms. Why would people bother investing this effort unless proof errors are common?

There are many easy answers to this question: for example, I *enjoy* explaining proofs to a computer. More to the point, various references above show that some people, typically extrapolating from anecdotes, say that proof errors are common. An observation that some people seem to be acting upon this belief is not independent evidence that proof errors are common.

2.6 Summary so far

One can understand [54], [55], [74], [62], [101], [75], [76], and papers on computer-checked proofs as suggesting that errors in algorithm analyses are common; but the evidence for this suggestion is weak.

On the other hand, the suggestion could be correct. Suggestions to the contrary are also based on weak evidence, as we will see in Section 3.

[7] The list of claimed errors is not provided in [76] for spot-checking; instead [76, last paragraph] invites the reader to redo the entire analysis.

3 Weak evidence that algorithm-analysis errors are rare

This section is analogous to Section 2 in considering weak evidence, but now the weaknesses are in evidence for suggestions that algorithm-analysis errors *rarely* occur.

3.1 Proofs are guarantees

It is easy to find literature describing "proofs" as guarantees of knowledge. Recall the "Proofs of security give an iron-clad guarantee" statement from [70] quoted in Section 1, for example; or consider Skiena's algorithm textbook [98, page 4] saying that "Correct algorithms usually come with a proof of correctness, which is an explanation of *why* we know that the algorithm must take every instance of the problem to the desired result".

It is easy to see how readers could end up thinking that claimed "proofs" are always correct—"100% certainty", as [74] puts it. The statements from [70] and [98] are not accompanied by warnings such as "The portion of the literature labeled as provable security does not give us iron-clad guarantees: some claimed proofs are wrong" or "We do not know that algorithms labeled as having proofs of correctness always work correctly: some claimed proofs are wrong, and sometimes the algorithms turn out not to work correctly".

Meanwhile readers who have separately heard about a few erroneous "proofs" can easily be led to think that errors are so rare as to not be worth mentioning. The book [70] states many theorems[8] and evidently endorses them. But consider the online list [71] of errata for [70]. The first entry in the list says "page 58, Theorem 3.11: f should be computable in polynomial time". This is not a clearly worded retraction, but inspecting [70, Theorem 3.11, "PROOF (Sketch)"] finds the problem: inside the proof sketch, there is a construction of an algorithm with a step that outputs "0 if and only if $\mathcal{A}$ outputs $f(m)$"; the rest of the proof does not work unless this algorithm takes polynomial time; justifying this requires an assumption that input-output pairs for f are recognizable in polynomial time; [70, Theorem 3.11] failed to make any such assumption.

It is easy to imagine that proofs included in a textbook are simpler and better reviewed than average proofs, especially once the textbook has reached its third edition; and yet a "theorem" claimed in [70] had to be withdrawn as unproven. These considerations do not tell us what the overall error rate is, but they do illustrate weaknesses in the evidence that [70] provides for a low error rate.

3.2 Known flaws are rare

Goldreich commented in 2007 [60] on "the unfortunate (and rare) cases in which flaws were found in published claimed 'proofs' (of security)". Goldreich continued by saying that these cases "reinforce the importance of careful verification of proofs, which constitute our only way of distinguishing facts from conjectures".

[8] The book [98] does not: it has a "strict no-theorem/proof policy".

In context, what this appears to be claiming is that normally proofs are carefully verified, but occasionally this important step is not taken before publication, so occasionally flaws are found in published proofs.

However, [60] provided no quantification, evidence, or citation for the claim that the known cases are "rare". It is easy to imagine reasons that someone can hear about a "proof of security" without hearing about the discovery of an error (for example, because of the error-suppression tendencies hypothesized in [62]), so one would guess that visibility bias will produce a perceived discovery rate below the actual discovery rate.

Furthermore, even if the rate of *known* errors in published proofs is low, the *actual* error rate of published proofs could be much higher. Consider the following numbers:

- Recall from Section 2.4 that [76] reported that Bergman's reviews of 84 papers had identified 28 errors classified by Lamport as "serious", including 11 false "theorem" statements.
- For comparison, [62, Figure 5] reported that papers in almost every area of mathematics had well below 1% chance of having an erratum issued by then.

A simple explanation for this gap is claimed in [76], namely that "Bergman was one of the few *Math Reviews* reviewers who read the papers carefully". This is consistent with the claims from, e.g., [54] and [101] about most papers not being checked carefully. A contrary claim in, e.g., [69] is that mistakes "rarely" happen; but [69] provides no evidence for this claim.

3.3 Mathematicians are careful

In 2007, Koblitz and Menezes [74] wrote the following:

> In theoretical mathematics, one of the reasons why theorems engender confidence and trust is that the proof of a major result is almost always scrutinized carefully by referees and others before publication.

However, the "almost always" claim is not quantified. Furthermore, the entire evidence presented or cited for this claim in [74] consists of two examples: (1) the discovery, by a referee, of a gap in the original Wiles "proof" of "Fermat's last theorem"; (2) the rapid confirmation at preprint time (see, e.g., [12] and [13]) of the Agrawal–Kayal–Saxena [5] theorem "PRIMES is in P".

There is no discussion in [74] of contrary examples. Consider, e.g., [83], a 1994 *Inventiones Mathematicae* paper by Nakamura and Uhlmann. This paper was withdrawn [84] in 2003 ("Unfortunately, we have not been able to prove the global result stated in [NU1]"), before the first version of [74] was posted. Does an article appearing in a top mathematics journal not qualify as a "major result"? If the concept of "major result" is so narrow, how is careful scrutiny of the occasional "major result" supposed to "engender confidence and trust" in mathematics more broadly?

Here are some newer examples. A 1999 *Annals of Mathematics* paper [6] by Alesker was withdrawn [7] in 2007: "We do not know if Theorem A is true." A

2006 *Inventiones Mathematicae* paper [40] by Biss and Farb was withdrawn [41] in 2009: "should be considered an open problem". See also [78] and [46].

The wiggle room of the word "almost" from [74] (regarding "theoretical mathematics"), like the wiggle room of the word "rare" from [60] (regarding "proofs of security"), allows any particular examples of errors in published proofs to be waved away. But extrapolating from some examples, and then inserting wiggle room to dismiss contrary examples, is not a valid basis for drawing conclusions. One could just as easily extrapolate from examples of incorrect proofs to the opposite claim—"almost all proofs are wrong"—and use the word "almost" to wave away any particular examples of correct proofs.

Section 2.2 mentioned the same paper [74] as questioning the extent to which "proofs of security" have been reviewed. The paper claims that "theoretical cryptography", unlike "theoretical mathematics", lacks a "tradition of careful examination of all important papers". Similarly, Koblitz in [73] claims that cryptographers "rarely read other authors' papers carefully". Perhaps this is true—but perhaps mathematicians also rarely read papers carefully. More to the point, perhaps algorithm analyses from all sources have a high error rate and are rarely checked carefully.

3.4 Actions, revisited

Section 2.5 mentioned that some papers report computer verification of proofs. Obviously many proof papers don't. On 23 January 2025, arXiv reported 106039 search results for "theorem" (including 10932 in the past 12 months); 93142 (including 10890) results for "proof"; 1028 (217) results for "formal verification"; 780 (139) results for "proof assistant"; 729 (102) results for "theorem prover"; 713 (96) results for "Coq" (the name of one of the available proof-checking tools, until its recent renaming as "Rocq"; skimming convinced me that many, although not all, of the results were for this tool rather than other things called "Coq"); 447 (101) results for "formally verified"; etc.

Papers *not* taking this step to ensure proof correctness can be interpreted as indications that the authors think proof errors (or at least their own proof errors) are rare. But this is weak evidence. People could instead be deterred by perceptions that the tools are too hard to use. Consider, e.g., Aaronson in 2019 [3] describing computer verification of proofs as "massive rote coding work that none of us has any intention of ever doing".

It is similarly weak evidence to point to examples of authors using theorems from other people without having checked the proofs, or specifically examples of authors relying on algorithm analyses from other people without having checked the analyses. While such examples could be viewed as expressing trust and suggesting that errors are too rare to worry about, they can also be explained by a perception that the cost of checking outweighs the benefits of doing so.

Finally, consider replacing "common" with "rare" at the end of Section 2.5. We have already seen people expressing a belief that proof errors are rare; an observation that some people seem to be acting upon this belief is not independent evidence that proof errors are rare.

4 How to confidently determine the error rate of a pool of papers with proofs

Here are two radically different hypotheses, neither of which is disproven by the weak evidence reviewed in Sections 2 and 3:

- The error rate of proof papers is below 1% (almost as small as the erratum rate reported in [62]). Reason: Proofs are almost always checked carefully. Invalid extrapolation from occasional failures sometimes leads people to wildly overestimate the error rate, to spread unfounded distrust in proofs, and to waste time having computers check proofs.
- The error rate of proof papers is above 50% (even higher than what [76] says is the rate of "serious" errors that Bergman pointed out in his reviews of selected algebra papers). Reason: People are overconfident and continually make mistakes. Persistent failure to check proofs leaves most proofs with undetected errors, also leading people to wildly underestimate the error rate.

This section gives reasons to believe that it is feasible to rule out at least one of these hypotheses, via a particular strategy to pin down the actual error rates of papers with proofs.

4.1 The strategy

The first step is obvious and easy: pick a random sample of (say) 100 papers to study. I'm not insisting on finding the *exact* error rates: statistical confidence is good enough for settling the risk-assessment debate.

For example, finding errors in 28 papers within a random sample of 84 papers (matching the non-random sample in [76]) would convincingly disprove the "below 1%" hypothesis. Indeed, if the hypothesis *is* correct then the chance of errors existing in ≥ 28 papers out of 84 is at most $\sum_{e \geq 28} \binom{84}{e} 0.01^e 0.99^{84-e} \approx 2^{-113}$; it's not plausible that a random experiment is so lucky. As another example, a sample of 100 papers having 100 correct papers would convincingly disprove the "above 50%" hypothesis. One can also easily calculate the "95% confidence interval" typically used in experimental sciences.

The hard part is to figure out how many of the papers in the sample are correct. It isn't sufficient here to follow the manual proof-checking steps that an author or referee would normally carry out: what we're trying to see is whether papers claiming "theorems" often have errors that are *not* caught by those steps. Saying "I've carried out a particularly diligent check of each proof step" or even "Look, I wrote a much more detailed presentation of the proof" still doesn't eliminate the question: maybe the detailed writeup has the same error as the original proof. The strategy, then, is to produce public *computer-checked* versions of the proofs, guaranteeing that the theorems really are correct.

This isn't the complete strategy yet: what happens if many papers in the sample have proof errors? The simple answer is to say "I don't believe the following proofs", but then a skeptical reader is left wondering whether the actual

situation is that the person who claims to have tried checking the proofs simply hasn't done enough work to recognize that the proofs are all fine. The strategy for convincingly settling such cases is to produce a public counterexample to a "proof" step.

There can be intermediate cases where someone reading a "proof" finds a gap in the logic—meaning a claim whose justification is unobvious to that reader—but at the same time finds no evident counterexamples to the claim. This could be a problem with the "proof", or a failure on the reader's part. Hopefully such cases (and cases of claims being ambiguous) can be rapidly resolved by communication with the "proof" authors; my experience is that sometimes a "proof" is withdrawn and sometimes a gap is filled. Note that if some papers in a sample end up with computer-checked proofs, while others end up with counterexamples, then leaving (say) 10% of papers unresolved won't have much effect on the statistical conclusions; on the other hand, if the error rate is actually very close to 0, then demonstrating this is incompatible with leaving so many proof gaps unfilled.

Conceptually, the rationale stated above for computer-checked proofs is not exactly the same as the rationale given in, e.g., [**101**]. The goal of [**101**] was to switch to computer-checked proofs to eliminate errors (compensating for a claimed problem regarding the current level of checking of proofs), whereas the goal of the strategy I'm describing is instead to convincingly determine the error rate. Perhaps the result will be that proof errors are extremely rare, in which case we can obtain high confidence in this via computer-checked proofs for a random sample, in much the same way that quality-assurance tests are applied to random samples of commercial products (see, e.g., [**82**]). Or perhaps the result will be *knowing* that proof errors are common, which would add weight to arguments for much broader usage of computer-checked proofs.

4.2 A pool of papers to study

The strategy from Section 4.1 doesn't care whether it's being applied to *all* papers with proofs. One can specify any pool of papers with proofs and then take a sample from that pool.

One reason for generalizing in this way is that there are interesting questions about variations across pools, such as whether proof reliability is changing over time (increasing because of improved access to preprints? decreasing because of increased publication pressure?) and whether peer review makes a big difference in error rates. Of course, determining error rates for multiple pools is a larger project than determining error rates for one pool, so it makes sense to start with one pool.

Another reason for not taking *all* papers with proofs is that, for the objective of understanding the error rate of algorithm analyses, extrapolating from the overall error rate of proofs could produce large errors in either direction. Consider again [**74**] suggesting that proofs in cryptography have higher error rates than proofs in "theoretical mathematics"; or consider the contrary hypothesis formulated in Section 2.4.

Furthermore, I want to try applying this strategy. My levels of interest and experience vary from one pool to another, so it makes sense for me to take a pool that I'm particularly interested in as an initial case study. I already have many papers on a wide range of algorithms to attack cryptographic systems—more than 60 such papers at last count, including some multi-algorithm analyses such as [14], [16], [8], [34], and [32]—so, okay, I'm taking the following pool: recent papers on algorithms to attack cryptographic systems.

To be concrete, let's define "recent" as papers that appeared in 2024 on the preprint server of IACR, the International Association of Cryptologic Research, not counting any subsequent revisions to the papers; I'm not limiting this to peer-reviewed papers. Here's a sampling mechanism: start the number-theory calculator `gp` and repeatedly call `random(2100)+1`, which produces a number between 1 and 2100; the papers on this preprint server in 2024 are numbered 1 through 2100. This calculator uses a deterministic seed for pseudorandomness by default, so anyone trying this will see the same sequence of paper numbers, namely 1679, 1473, 1498, etc. Skip the following papers: duplicates; papers that don't analyze attack algorithms against cryptographic systems;[9] and, to avoid conflict-of-interest questions, papers from anyone who (co)authored any of my papers since 2020. Take a total of 100 papers.

One reason I find this particular algorithmic topic intriguing is that it inherently involves considering the capabilities of computer resources far beyond the resources that we have available for tests. The whole point is to help users select safe cryptosystems; obviously users won't use a cryptosystem if *we* can demonstrate an attack breaking the cryptosystem, but we also want users to avoid the larger set of cryptosystems breakable by a real attacker with many billions of dollars of computer equipment. Attackers are also intercepting encrypted data in the hopes of decrypting it in the future using even larger computers or quantum computers; see, e.g., [63] and [89].

4.3 Won't this be an insane amount of work?

For people who have heard that writing a computer-checked proof is time-consuming—certainly worth a paper by itself, beyond the paper that presented the original proof—the idea of writing computer-checked proofs for 100 papers might sound like a lifetime of work, so the only way to produce statistically meaningful results within (say) 5 years would be to split the work across many people. I have four reasons to think that, no, this is within reach for one person.

First, as soon as I've found a counterexample to any claim in any algorithm analysis in a paper, that paper goes into the "error" column and I don't have to spend more time on it. I expect this to happen frequently. This isn't cheating: *one* false claim in a "proof" means that the "proof" is not correct. Presumably some of these "proofs" will be fixable (maybe very easily fixable), but that is not relevant

[9] Note to any cryptographers reading this: I mean the mathematical systems, not the implementations, so side-channel attacks aren't included here.

to the question of whether the paper proved what it claimed. Counting only "serious" errors, as in [76], could be another topic of interest but is subjective, dependent upon who's checking proofs, as [76] notes; see also the fascinating way that the word "typo" was used in the retraction analyzed in [19, Section 6.5]. Furthermore, presumably cases where people agree that a "theorem" is easily fixable have an overlap with cases where the non-fixed "theorem" leads to disaster, such as the XCB incident reviewed in Section 1 (or the "OCB2" incident; see [68]). We want theorems to be trustworthy as stated, not just to make an intellectual contribution.

Second, while an algorithm analysis can occupy many pages, it can also be a small component of a paper, reducing the amount of work correspondingly. Also, recall from Section 4.1 that occasionally skipping a hard-to-evaluate paper is okay when error rates aren't very close to 0% (or 100%).

Third, the time I needed to write computer-checked proofs for four of my recent papers (see [21], [28], [24], and [27]) was just a few weeks per paper. For one paper, I explained the theorems to the Lean proof-checker, and then explained them all again to the HOL Light proof-checker; I used HOL Light for everything else. See my report [26] for further information about the time consumption. These are papers where I already knew exactly how the proofs worked; to test whether that really matters, I recently wrote a computer-checked proof [30] of the transcendence of π, and then [31] of the full Lindemann theorem (often called Lindemann–Weierstrass), in each case taking just a few weeks.

Fourth, most of my time in writing computer-checked proofs has been spent proving background lemmas, and I've already been able to productively reuse some lemmas across my HOL Light projects.[10] Even without any other improvements in proof-checking tools, I won't be surprised to see further reuse of lemmas reduce my average computer-checking time below a week per checked paper. For papers with counterexamples, I'm hoping for under a day per paper.

A caveat covered in [26] is that in the meantime I had two papers with proofs that I *didn't* try explaining to a proof-checker, in both cases because I estimated that proving the relevant background would take a few months of work. In particular, it's critical to have a broad enough definition of "algorithm"; see Section 4.4. This work will be shared across many theorems, and looking at the proof techniques used in the papers selected in Section 4.2 makes me believe that *for this pool* there won't be other gaps at this level of importance.[11]

[10] I plan to continue using HOL Light. My time estimates are based on this. The reasons that I selected HOL Light for my previous proof-checking projects certainly don't guarantee that HOL Light is the most efficient option overall, and I've also been investigating other tools (see, e.g., [26, Section 4]); but, for checking algorithm-analysis proofs, I have enough confidence in feasibility using HOL Light that the possibility of saving time using something else is outweighed by the possibility of losing time using something else.

[11] For comparison, I would expect applying the same strategy to a sample from *all* proof papers to be more difficult—still feasible, but I would want to spread the work out more, for example delegating differential-geometry proofs to a geometer.

Just in case I'm underestimating the total work, I plan to start with the first 6 papers[12] in the sequence described in Section 4.2; then expand to the first 12; then expand to the first 25; then expand to the first 50; and finally cover all 100. But I don't expect this precaution to be necessary.

4.4 Defining algorithms

In 2020, Forster, Kunze, and Wuttke [57] provided various computer-checked proofs (in Coq) starting with a general definition of deterministic multi-tape Turing machines. This might seem at first glance to be an adequate foundation for proving typical theorems about algorithms. One proof says that a particular deterministic multi-tape Turing machine needs $22m+12n+98$ steps to compute the sum of an m-bit integer and an n-bit integer.

However, the literature defines and uses many concepts of "algorithm" beyond deterministic multi-tape Turing machines. Some of these definitional variations aren't a big deal to skip, but some are essential for the task of verifying existing algorithm analyses.

As an example of something that's commonly done but easily skipped, consider extending the definition of Turing machines to drop the requirement of determinism. This extension allows "do this" and "do that" instructions to be combined into "do this or do that"; in particular, there is then a combined instruction "store 0 or store 1", i.e., a coin flip. This extension is commonly used to define non-deterministic complexity classes such as NP and randomized complexity classes such as RP and BPP. To skip this extension, simply provide a sequence of coin flips as another input to a deterministic machine. This means, e.g., taking [86, Proposition 9.1] as the definition of NP rather than proving it as a consequence of a non-deterministic definition. Proofs using NP normally use this characterization anyway, as admitted in [86, page 425].

Here's an example of something that's clearly *not* skippable. While the literature sometimes tracks every bit operation in a large computation (see, e.g., my recent paper [33] with Chou) and sometimes tracks data layout on a multi-tape Turing machine (see, e.g., the 1994 book [94] from Schönhage, Grotefeld, and Vetter), it is easier—and much more common among the papers I've read—to allow larger operations, for example allowing $c \leftarrow a + b$ and $c \leftarrow x[i]$ as single instructions. Verifying a theorem regarding the number of such instructions requires definitions that allow these instructions inside algorithms.

As a side note, such instruction counts can be a remarkably poor predictor of real-world cost. See, e.g., Shamir's 1977 algorithm [95] to find a nontrivial factor of a composite integer $n \geq 4$ using $O(\log n)$ arithmetic instructions.[13] But

[12] I know from experience that it's good for me to have multiple projects to work on at once, so I don't want to force myself to finish handling each paper before looking at the next. On the other hand, I handled the first paper already; see Section 5.1.

[13] The algorithm begins with $O(\log n)$ multiplications to compute $(2^n + 1)^e$ for a selected value of $e \in \Theta(n)$. The literature sometimes avoids such abuses by prohibiting multiplication instructions, as in [86, Sections 2.6 and 15.2], but this

the question of whether the metric chosen for an algorithm analysis is realistic is different from the question of whether the analysis is correct.

Another common extension is a "call an oracle" instruction. The function computed by this instruction is a parameter that can be specified later, and theorems often consider multiple possibilities for this parameter. As one of many examples, my recent not-yet-computer-checked paper [**32**, Theorem 2.5.1] puts an upper bound on the success probability for a specified task of any algorithm making "at most q distinct calls to the F oracle", where F is defined as "a uniform random injective function from $\{0,1\}^{b+h}$ to $\{0,1\}^c$", assuming "$c \geq b + h$"; success probability is "by definition averaged over all choices of the oracle".

The cost of an algorithm, like the correctness of an algorithm, is often a random variable. As a simple illustration, one of the aforementioned algorithm textbooks [**52**, Section 5.4.2] describes a process of tossing balls independently and uniformly at random into bins $1, 2, \ldots, b$; [**52**] asks questions such as "How many balls must you toss, on the average, until a given bin contains a ball?" and "How many balls must you toss until every bin contains at least one ball?". Note that the set of possible ball-tossing runs is uncountable except for small b, even if "until" terminates a run;[14] the standard mathematical definition of probabilities on such sets involves measure theory[15] at the level reviewed in, e.g., my paper [**11**, Appendix]. See [**67**] for a formal development of probabilistic algorithms, including measure theory, in the HOL4 proof-checker. For HOL Light, the theory of Lebesgue measure on $\mathbb{R}^d$ is available, and then one can simply take the uniform distribution on $\{x \in \mathbb{R} : 0 \leq x < 1\}$ as a source of random bits via binary expansions, ignoring the measure-0 set $\{s \in \{0,1\}^\infty : \#\{i : s_i = 0\} < \infty\}$ of strings that cannot occur as expansions.

A general theory of algorithm execution on top of any set of instructions is already available in HOL Light (see [**81**]) and supports statements such as "this program terminates after this number of steps, producing an output satisfying this property". This theory is usable: consider, e.g., the computer-checked proof from [**64**] that a particular sequence of Intel instructions correctly computes scalar multiplication on Curve25519, an elliptic curve that I had introduced in [**15**]. The Intel instruction set operates on bounded-size states, but substituting another instruction set will allow statements such as "this algorithm takes polynomial time", and including an oracle-call instruction will allow statements such as "this algorithm uses at most q oracle calls". I plan to build statements about algorithm executions on random inputs as a modular layer on top of that.

---

still allows a linear number of instructions to generate a quadratic number of bits. A more direct fix is to have a parameter—typically growing logarithmically with the input size—that limits the number of bits in each integer; see, e.g., [**38**, Section 6].

[14] Consider, e.g., the set of possible ball-tossing sequences, terminated as soon as bin 1 contains a ball. This set includes $\{2,3\}^\infty$ if $b \geq 3$.

[15] Various proof gaps in the literature seem easy to explain from the fact that measure theory is not part of the standard computer-science curriculum. For example, [**52**, Appendix C.3] says that uncountable sample spaces "are unnecessary to address for our purposes", and [**9**] says that a uniform random function from an infinite set to a finite set "has no meaning".

5 Beyond proofs

So far I've been describing algorithm analysis as a sub-topic of proofs. The reality is more complicated: an algorithm analysis can have some proven components and some heuristic components, so the analysis can be incorrect even if all of the proofs are correct. Sometimes an analysis that relies on heuristics is mislabeled as a proven analysis.

Section 5.1 presents a recent example of this phenomenon. Section 5.2 fits this example into the context of various examples identified in my paper [**33**] with Chou. Section 5.3 extends the strategy from Section 4.1 to cover algorithm analyses beyond proofs.

5.1 Case study: paper 1679

Let's look at the first paper in the sequence from Section 4.2, namely [**47**]. This paper is in the specified pool: it's a paper on "information-set decoding" (ISD), which is the fastest technique known to attack various cryptosystems based on error-correcting codes. Skimming the paper finds, among other things, [**47**, Corollary 4.1] giving a formula for the "asymptotic average complexity of Algorithm 1".

[**47**, Algorithm 1] says that its inputs are "$H \in \mathcal{R}^{(n-k_0)\times n}$", "$s \in \mathcal{R}^{n-k_0}$", "$w \in \mathbb{N}$", and "$v \leq \min\{M \cdot K, w\}$"; and that its output is a vector "$e \in \mathcal{R}^n$ such that $\mathrm{wt_d}(e) \leq w$ and $He^\top = s^\top$".

Here $\mathcal{R}$ is any "Galois ring"; [**47**, Example 2.1] says that any prime field is a Galois ring, so let's focus on $\mathcal{R} = \mathbb{Z}/2$ for concreteness. The quantity K is the minimum cardinality of a subset of C that generates C as an $\mathcal{R}$-module, where C is the kernel of H; see [**47**, Definitions 2.3 and 2.4]. The quantity M is the maximum "weight" of any length-1 vector. In the context of [**47**, Corollary 4.1], "weight" and "$\mathrm{wt_d}$" both refer to Hamming weight (number of nonzero entries), so $M = 1$.

Inside the algorithm, the only vectors e that can be returned satisfy "$e_I = e_1$" (line 6), meaning that the restriction of e to a particular set I of coordinates is e_1. Also "$e_1 \in P$" (line 4), which in turn implies "$\mathrm{wt_d}(e_1) = v$" (line 3); i.e., any returned vector e has weight v on I. The set I is chosen as an "information set" (line 1), meaning that $\#I = K$ and $\#C_I = \#C$; see [**47**, Definition 2.6]. If we can find an input where these conditions can't be satisfied, then the algorithm will loop forever (line 7).

Let's try a simple example: H maps $\mathcal{R}^n$ to $\mathcal{R}^{n-k_0}$ by removing the last k_0 coordinates. The kernel $C \subseteq \mathcal{R}^n$ consists of all vectors supported on the last k_0 coordinates. The rank K of C is k_0. A set I is an information set exactly when it consists of the last k_0 coordinates, so a vector e returned by this algorithm must have weight v on the last k_0 coordinates.

Take specifically $k_0 = \lfloor n/2 \rfloor$, and assume $n \geq 2$; then $0 < k_0 < n$. Consider, as algorithm input, this matrix H; the vector $s = (1, 0, 0, \ldots, 0) \in \mathcal{R}^{n-k_0}$; $w = 1$; and $v = 1$, which satisfies $v \leq MK$ (since $1 \leq k_0 = K$) and $v \leq w$ as required. A correct algorithm would return a vector $e \in \mathcal{R}^n$ of weight ≤ 1 that maps to

s, i.e., a vector of weight ≤ 1 whose first $n - k_0$ coordinates are $(1, 0, 0, \ldots, 0)$. This vector must be $(1, 0, 0, \ldots, 0) \in \mathcal{R}^n$, which has weight 0 on the last k_0 coordinates, so it cannot be returned by [47, Algorithm 1], so the algorithm runs forever.

A statement about "average" cost could mean that, for *each* algorithm input, the algorithm has the stated cost on average over coin flips inside the algorithm; or it could also be averaging over all finitely many (H, s) for this (n, k_0, v, w). Either way, having one input running forever means that the average is infinite. One can check that the formula in [47, Corollary 4.1] is finite.

From the perspective of the strategy described in Section 4.1 as applied to the pool described in Section 4.2, this counterexample disposes of [47] (leaving 99 further papers to investigate from this sample), since it is a counterexample to something the paper claims to have proven. This counterexample took under a day to find and check, in line with the predictions from Section 4.3.

5.2 Heuristic algorithm analyses

Wait a minute: [47] says that various earlier papers already introduced and analyzed these ISD algorithms for the case of fields $\mathcal{R}$. For example, [47] credits [47, Algorithm 1] to a 1988 paper [77] by Lee and Brickell. More than 50 ISD papers are listed in [29]. The counterexample from Section 5.1 takes a field $\mathcal{R} = \mathbb{Z}/2$. Does this mean that every other ISD paper we look at will also have a wrong proof?

Answer: No. For example, the Lee–Brickell paper [77] *doesn't claim any theorems*. A 1989 paper by Stern [100] states theorems regarding an improved algorithm, but notes that those theorems are regarding a heuristic *model* of the algorithm performance, a model discounting possible (anti-)correlations between outputs and information sets: "We think that it is a reasonable measure of the probability ... although it neglects the role of the possible revisions in the choices of the columns that are performed during step 1." Similar disclaimers appear in some newer ISD papers, such as [35, Section 3, "bias"] and [36, Section 5].

It's easy to see that almost all large matrices H have many more information sets than the example in Section 5.1; [50] plausibly claims to be able to prove that ISD performance for "virtually all" choices of H has the same asymptotic exponents as the model. However, for code-based cryptography, we care about the matrices H that appear in various code-based cryptosystems. Small-scale experiments haven't found ISD distinguishing this class of matrices from most matrices, nor does anyone have an explanation of how that could happen—but nobody has proven ISD performance for this class of matrices; it's just a conjecture based on a heuristic. No algorithm for these matrices has been *proven* to be as fast as the conjectured performance of ISD.

Will someone who wants to get a computation done ignore what seems to be the fastest algorithm, merely because there's no proof that it's the fastest? Possibly—the user might consider speedups less important than being sure in advance what the performance will be—but surely many users will decide to make the opposite tradeoff, especially when the unproven speedups are large.

This tension between proofs and performance is not unique to ISD. [33, Appendix B] surveys unproven speedups for integer factorization, elliptic-curve discrete logarithms, and lattice problems, and explains why this should not be surprising—even if it is different from the impression conveyed by the algorithms selected for typical textbooks.

5.3 High assurance for algorithm analyses beyond proofs

One can, with enough effort, increase the level of assurance of a claimed proof of the cost and success probability of an algorithm: explain the proof to a computer. When proofs are not available, can one increase the level of assurance of an *unproven* analysis of the cost and success probability of an algorithm?

The main point of [33] is to provide an answer to this question, along with demos of the answer. The answer starts the same way as a computer-checked proof: one has to write

- a complete definition of the model of computation including a cost metric,
- a complete definition of the problem supposedly being solved,
- a complete definition of the algorithm in this model of computation,
- a complete formula for the predicted cost of the algorithm, and
- a complete formula for the predicted success probability of the algorithm

in a language that a computer can understand. But the dénouement is different:

- Given a proof of the predictions, one would finish by explaining the proof to the computer and having the computer verify the proof, as in Section 4.1.
- The approach of [33] is instead to have the computer simulate the algorithm on a sample of inputs, comparing the observed cost and the observed success probability to the predictions.

The demos in [33] are for a simple brute-force search and a spectrum of ISD algorithms, in both cases with a perfect match of observed cost to predictions and a close match of observed success probability to predictions. There are various examples in [33] of how this structure—with its computer-enforced links between the model of computation, the cost metric, the algorithm definitions, the predictions, and the simulations—would have caught errors in unproven algorithm analyses in the literature.

So it is natural to consider the following supplement to the strategy from Section 4.1: instead of ignoring the components of an algorithm analysis that are not labeled as proofs,[16] try fitting the algorithm analysis into the structure from [33]. Perhaps this will catch an error; perhaps it will observe a close match between predictions and simulations. A close match is not as convincing as a computer-checked proof—for example, perhaps an analysis error becomes invisible when one scales down input sizes to be able to afford simulations—but

[16] If there is an error in something that *is* labeled as a proof then this supplement does not apply. For example, [47] is still classified as "counterexample to claimed theorem". Again, the point is to determine whether we can trust what papers claim.

it is still interesting to see how often algorithm analyses survive this level of scrutiny.

The central software framework from [33] expresses all algorithms as compositions of bit operations. Bit operations are how real chips carry out computations, but it is desirable to support more cost metrics so as to check analyses of those metrics, and it is desirable to match definitions to Section 4.4 to be able to connect simulations to any available proofs. Also, currently the only success notion supported by the framework is "the algorithm found $f(x)$ given $g(x)$"; I plan to extend the framework to support a wider range of specifications of what it means for an algorithm to succeed.

Finally, "trapdoor simulation" from [18] can, in at least some cases, efficiently simulate interesting quantum circuits, without having a quantum computer available for simulations. However, sampling makes me think that, for the occasional quantum circuits in the pool from Section 4.2, it suffices to focus on computer-checked proofs. Note that quantum circuits aren't hard to define; see, e.g., [44] or my exposition [20].

References

[1] — (no editor), *37th annual symposium on foundations of computer science, FOCS '96, Burlington, Vermont, USA, October 14–16, 1996*, Institute of Electrical and Electronics Engineers, 1996. ISBN 0-8186-7594-2. URL: https://ieeexplore.ieee.org/xpl/conhome/4141/proceeding. See [9].

[2] — (no editor), *12th IEEE international conference on automatic face & gesture recognition, FG 2017, Washington, DC, USA, May 30–June 3, 2017*, IEEE Computer Society, 2017. ISBN 978-1-5090-4023-0. URL: https://ieeexplore.ieee.org/xpl/conhome/7959734/proceeding. See [87].

[3] Scott Aaronson, *Death of proof greatly exaggerated* (2019). URL: https://scottaaronson.blog/?p=4133. Citations in this document: §3.4.

[4] Carlisle M. Adams, Ali Miri, Michael J. Wiener (editors), *Selected areas in cryptography, 14th international workshop, SAC 2007, Ottawa, Canada, August 16–17, 2007, revised selected papers*, 4876, Springer, 2007. ISBN 978-3-540-77359-7. DOI: 10.1007/978-3-540-77360-3. See [80].

[5] Manindra Agrawal, Neeraj Kayal, Nitin Saxena, *PRIMES is in P*, Annals of Mathematics. Second Series **160** (2004), 781–793. URL: https://annals.math.princeton.edu/wp-content/uploads/annals-v160-n2-p12.pdf. DOI: 10.4007/annals.2004.160.781. Citations in this document: §3.3.

[6] Semyon Alesker, *Continuous rotation invariant valuations on convex sets*, Annals of Mathematics, Second Series **149** (1999), 977–1005; see also newer version [7]. URL: https://eudml.org/doc/120431. DOI: 10.2307/121078. Citations in this document: §3.3.

[7] Semyon Alesker, *Erratum: Rotation invariant valuations on convex sets*, Annals of Mathematics, Second Series **166** (2007), 947–948; see also older version [6]. URL: https://annals.math.princeton.edu/wp-content/uploads/annals-v166-n3-p08.pdf. DOI: 10.4007/annals.2007.166.947. Citations in this document: §3.3.

[8] Mihir Bellare, Daniel J. Bernstein, Stefano Tessaro, *Hash-function based PRFs: AMAC and its multi-user security*, in Eurocrypt 2016 [56] (2016), 566–595. URL:

https://cr.yp.to/papers.html#amac. DOI: 10.1007/978-3-662-49890-3_22. Citations in this document: §4.2.

[9] Mihir Bellare, Ran Canetti, Hugo Krawczyk, *Pseudorandom functions revisited: the cascade construction and its concrete security*, in FOCS 1996 [**1**] (1996), 514–523. URL: https://www-cse.ucsd.edu/~mihir/papers/cascade.html. DOI: 10.1109/SFCS.1996.548510. Citations in this document: §4.4.

[10] Mihir Bellare, Phillip Rogaway, *Optimal asymmetric encryption*, in Eurocrypt 1994 [**93**] (1994), 92–111. URL: https://cseweb.ucsd.edu/~mihir/papers/oaep.pdf. DOI: 10.1007/BFb0053428. Citations in this document: §2.2.

[11] Daniel J. Bernstein, *How to stretch random functions: the security of protected counter sums*, Journal of Cryptology **12** (1999), 185–192. URL: https://cr.yp.to/papers.html#stretch. DOI: 10.1007/S001459900051. Citations in this document: §4.4.

[12] Daniel J. Bernstein, *Proving primality after Agrawal-Kayal-Saxena* (2003). URL: https://cr.yp.to/papers.html#aks. Citations in this document: §3.3.

[13] Daniel J. Bernstein, *Computing logarithm floors in essentially linear time* (2003). URL: https://cr.yp.to/papers.html#logfloor. Citations in this document: §3.3.

[14] Daniel J. Bernstein, *Stronger security bounds for Wegman-Carter-Shoup authenticators*, in Eurocrypt 2005 [**53**] (2005), 164–180. URL: https://cr.yp.to/papers.html#securitywcs. DOI: 10.1007/11426639_10. Citations in this document: §4.2.

[15] Daniel J. Bernstein, *Curve25519: new Diffie-Hellman speed records*, in PKC 2006 [**103**] (2006), 207–228. URL: https://cr.yp.to/papers.html#curve25519. DOI: 10.1007/11745853_14. Citations in this document: §4.4.

[16] Daniel J. Bernstein, *Proving tight security for Rabin-Williams signatures*, in Eurocrypt 2008 [**99**] (2008), 70–87. URL: https://cr.yp.to/papers.html#rwtight. DOI: 10.1007/978-3-540-78967-3_5. Citations in this document: §2.2, §4.2.

[17] Daniel J. Bernstein, *Multi-user Schnorr security, revisited* (2015). URL: https://cr.yp.to/papers.html#multischnorr. Citations in this document: §2.2.

[18] Daniel J. Bernstein, *Trapdoor simulation of quantum algorithms* (2015). URL: https://cr.yp.to/talks/2015.04.03/slides-djb-20150403-a4.pdf. Citations in this document: §5.3.

[19] Daniel J. Bernstein, *Comparing proofs of security for lattice-based encryption* (2019), Second PQC Standardization Conference. URL: https://cr.yp.to/papers.html#latticeproofs. Citations in this document: §2.2, §4.3.

[20] Daniel J. Bernstein, *Quantum algorithms* (2019). URL: https://cr.yp.to/talks.html#2019.07.01-1. Citations in this document: §5.3.

[21] Daniel J. Bernstein, *Verified fast formulas for control bits for permutation networks* (2020). URL: https://cr.yp.to/papers.html#controlbits. Citations in this document: §4.3.

[22] Daniel J. Bernstein, *Skeptical about Corollary 7 of https://arxiv.org/abs/2110.13352. Doesn't Theorem 6 need to assume that gamma is bounded away from 1, which in turn needs N to have a higher exponent? Might still beat other approaches but needs more careful analysis of the run time, heuristic accuracy, etc.* (2021). URL: https://twitter.com/hashbreaker/status/1459198701826547713. Citations in this document: §1.

[23] Daniel J. Bernstein, *Looking a bit more at https://arxiv.org/abs/2110.13352, and now skeptical about Theorem 6 (never mind the application to Corollary 7). The last proof step says 2^t amplifications each costing sqrt(N), but aren't half of these nested amplifications? How are further sqrt(N) factors avoided?* (2021). URL: https://twitter.com/hashbreaker/status/1469253334418997256. Citations in this document: §1.

[24] Daniel J. Bernstein, *Hull Light* (2023). URL: https://cr.yp.to/2023/hull-light-20230416.sage. Citations in this document: §4.3.

[25] Daniel J. Bernstein, *Multi-ciphertext security degradation for lattices* (2023). URL: https://cr.yp.to/papers.html#lprrr. Citations in this document: §2.2.

[26] Daniel J. Bernstein, *Papers with computer-checked proofs* (2024). URL: https://cr.yp.to/papers.html#pwccp. Citations in this document: §4.3, §4.3, §4.3.

[27] Daniel J. Bernstein, *Understanding binary-Goppa decoding*, IACR Communications in Cryptology **1** (2024), article 1.14. URL: https://cr.yp.to/papers.html#goppadecoding. DOI: 10.62056/ANGY4FE-3. Citations in this document: §2.5, §4.3.

[28] Daniel J. Bernstein, *Asymptotics for the standard block size in primal lattice attacks: second order, formally verified*, IACR Communications in Cryptology (2024), to appear. URL: https://cr.yp.to/papers.html#latticeasymp. Citations in this document: §4.3.

[29] Daniel J. Bernstein, *Information-set decoding* (2024). URL: https://isd.mceliece.org/. Citations in this document: §5.2.

[30] Daniel J. Bernstein, *transcendence-pi-20241122* (2024). URL: https://cr.yp.to/2024/transcendence-pi-20241122.ml. Citations in this document: §4.3.

[31] Daniel J. Bernstein, *transcendence-20241221* (2024). URL: https://cr.yp.to/2024/transcendence-20241221.ml. Citations in this document: §4.3.

[32] Daniel J. Bernstein, *FO derandomization sometimes damages security*, IACR Communications in Cryptology (2024), to appear. URL: https://cr.yp.to/papers.html#footloose. Citations in this document: §4.2, §4.4.

[33] Daniel J. Bernstein, Tung Chou, *CryptAttackTester: high-assurance attack analysis*, in Crypto 2024 [**88**] (2024), 141–182. URL: https://cr.yp.to/papers.html#cat. DOI: 10.1007/978-3-031-68391-6_5. Citations in this document: §4.4, §5, §5.2, §5.3, §5.3, §5.3, §5.3, §5.3, §5.3.

[34] Daniel J. Bernstein, Andreas Hülsing, *Decisional second-preimage resistance: when does SPR imply PRE?*, in Asiacrypt 2019 [**59**] (2019), 33–62. URL: https://cr.yp.to/papers.html#dspr. DOI: 10.1007/978-3-030-34618-8_2. Citations in this document: §4.2.

[35] Daniel J. Bernstein, Tanja Lange, Christiane Peters, *Attacking and defending the McEliece cryptosystem*, in PQCrypto 2008 [**45**] (2008), 31–46. URL: https://cr.yp.to/papers.html#mceliece. DOI: 10.1007/978-3-540-88403-3_3. Citations in this document: §5.2.

[36] Daniel J. Bernstein, Tanja Lange, Christiane Peters, *Smaller decoding exponents: ball-collision decoding*, in Crypto 2011 [**91**] (2011), 743–760. URL: https://cr.yp.to/papers.html#ballcoll. DOI: 10.1007/978-3-642-22792-9_42. Citations in this document: §5.2.

[37] Daniel J. Bernstein, Edoardo Persichetti, *Towards KEM unification* (2018). URL: https://cr.yp.to/papers.html#tightkem. Citations in this document: §2.2.

[38] Daniel J. Bernstein, Jonathan P. Sorenson, *Modular exponentiation via the explicit Chinese remainder theorem*, Mathematics of Computation **76** (2007),

443–454. URL: https://cr.yp.to/papers.html#meecrt. DOI: 10.1090/S0025-5718-06-01849-7. Citations in this document: §4.4.

[39] Amit Singh Bhati, Michiel Verbauwhede, Elena Andreeva, *Breaking, repairing and enhancing XCBv2 into the tweakable enciphering mode GEM* (2024). URL: https://eprint.iacr.org/2024/1554. Citations in this document: §1, §1, §1.

[40] Daniel Biss, Benson Farb, *$\mathcal{K}_g$ is not finitely generated*, Inventiones Mathematicae **163** (2006), 213–226; see also newer version [**41**]. URL: https://arxiv.org/abs/math/0405386. DOI: 10.1007/s00222-005-0472-x. Citations in this document: §3.3.

[41] Daniel Biss, Benson Farb, *Erratum: $\mathcal{K}_g$ is not finitely generated*, Inventiones Mathematicae **178** (2009), 229–229; see also older version [**40**]. DOI: 10.1007/s00222-009-0202-x. Citations in this document: §3.3.

[42] Jasmin Blanchette, Catalin Hritcu (editors), *Proceedings of the 9th ACM SIGPLAN international conference on certified programs and proofs, CPP 2020, New Orleans, LA, USA, January 20–21, 2020*, ACM, 2020. ISBN 978-1-4503-7097-4. See [57].

[43] Alexandra Boldyreva, Daniele Micciancio (editors), *Advances in cryptology—CRYPTO 2019—39th annual international cryptology conference, Santa Barbara, CA, USA, August 18–22, 2019, proceedings, part I*, 11692, Springer, 2019. ISBN 978-3-030-26947-0. DOI: 10.1007/978-3-030-26948-7. See [68].

[44] Anthony Bordg, Hanna Lachnitt, Yijun He, *Certified quantum computation in Isabelle/HOL*, Journal of Automated Reasoning **65** (2021), 691–709. URL: https://core.ac.uk/download/pdf/429667276.pdf. DOI: 10.1007/s10817-020-09584-7. Citations in this document: §5.3.

[45] Johannes Buchmann, Jintai Ding (editors), *Post-quantum cryptography, second international workshop, PQCrypto 2008, Cincinnati, OH, USA, October 17–19, 2008, proceedings*, 5299, Springer, 2008. ISBN 978-3-540-88402-6. DOI: 10.1007/978-3-540-88403-3. See [35].

[46] Kevin Buzzard, *Fermat's Last Theorem — how it's going* (2024). URL: https://xenaproject.wordpress.com/2024/12/11/fermats-last-theorem-how-its-going/. Citations in this document: §3.3.

[47] Giulia Cavicchioni, Alessio Meneghetti, Giovanni Tognolini, *Information set decoding for ring-linear code* (2024). URL: https://eprint.iacr.org/2024/1679. Citations in this document: §5.1, §5.1, §5.1, §5.1, §5.1, §5.1, §5.1, §5.1, §5.1, §5.1, §5.2, §5.2, §5.2, §5.3.

[48] Debrup Chakraborty, Vicente Hernandez-Jimenez, Palash Sarkar, *Another look at XCB*, Cryptography and Communications **7** (2015), 439–468. URL: https://eprint.iacr.org/2013/823. DOI: 10.1007/s12095-015-0127-8. Citations in this document: §1, §1, §1, §1, §1.

[49] Yilei Chen, *Quantum algorithms for lattice problems* (2024). URL: https://eprint.iacr.org/2024/555. Citations in this document: §1, §1.

[50] John T. Coffey, Rodney M. Goodman, *The complexity of information set decoding*, IEEE Transactions on Information Theory **35** (1990), 1031–1037. DOI: 10.1109/18.57202. Citations in this document: §5.2.

[51] Gérard D. Cohen, Jacques Wolfmann (editors), *Coding theory and applications, 3rd international colloquium, Toulon, France, November 2–4, 1988, proceedings*, Lecture Notes in Computer Science, 388, Springer, 1989. ISBN 0-387-51643-3. DOI: 10.1007/BFb0019841. See [100].

[52] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to algorithms*, 4th edition, MIT Press, 2022. ISBN 978-0-262-04630-5. Citations in this document: §1, §1, §4.4, §4.4, §4.4.

[53] Ronald Cramer (editor), *Advances in cryptology—EUROCRYPT 2005, 24th annual international conference on the theory and applications of cryptographic techniques, Aarhus, Denmark, May 22–26, 2005, proceedings*, 3494, Springer, 2005. ISBN 3-540-25910-4. DOI: 10.1007/b136415. See [14].

[54] Philip J. Davis, *Fidelity in mathematical discourse: is one and one really two?*, American Mathematical Monthly **79** (1972), 252–263. URL: https://gwern.net/doc/math/1972-davis.pdf. DOI: 10.1080/00029890.1972.11993025. Citations in this document: §2.2, §2.6, §3.2.

[55] Peter van Emde Boas, *The correspondence between Donald E. Knuth and Peter van Emde Boas on priority deques during the spring of 1977* (2013). URL: https://staff.fnwi.uva.nl/p.vanemdeboas/knuthnote.pdf. Citations in this document: §2.1, §2.6.

[56] Marc Fischlin, Jean-Sébastien Coron (editors), *Advances in cryptology—EUROCRYPT 2016—35th annual international conference on the theory and applications of cryptographic techniques, Vienna, Austria, May 8–12, 2016, proceedings, part I*, 9665, Springer, 2016. ISBN 978-3-662-49889-7. DOI: 10.1007/978-3-662-49890-3. See [8].

[57] Yannick Forster, Fabian Kunze, Maxi Wuttke, *Verified programming of Turing machines in Coq*, in CPP 2020 [**42**] (2020), 114–128. URL: https://www.ps.uni-saarland.de/Publications/documents/ForsterEtAl_2019_VerifiedTMs.pdf. DOI: 10.1145/3372885.3373816. Citations in this document: §4.4.

[58] Christoph G. Günther, *Advances in cryptology—EUROCRYPT '88, proceedings of the workshop on the theory and application of cryptographic techniques held in Davos, May 25–27, 1988*, Lecture Notes in Computer Science, 330, Springer, 1988. ISBN 3-540-50251-3. DOI: 10.1007/3-540-45961-8. See [77].

[59] Steven D. Galbraith, Shiho Moriai (editors), *Advances in cryptology—ASIACRYPT 2019—25th international conference on the theory and application of cryptology and information security, Kobe, Japan, December 8–12, 2019, proceedings, part III*, 11923, Springer, 2019. ISBN 978-3-030-34617-1. DOI: 10.1007/978-3-030-34618-8. See [34].

[60] Oded Goldreich, *Koblitz article misleading*, Notices of the AMS **54** (2007), 1454–1454. URL: https://www.ams.org/notices/200711/tx071101454p.pdf. Citations in this document: §3.2, §3.2, §3.3.

[61] W. T. Gowers, *The two cultures of mathematics* (2007). URL: https://www.dpmms.cam.ac.uk/~wtg10/2cultures.pdf. Citations in this document: §2.4.

[62] Joseph F. Grcar, *Errors and corrections in mathematics literature*, Notices of the AMS **60** (2013), 418–425. URL: https://www.ams.org/notices/201304/rnoti-p418.pdf. DOI: 10.1090/noti988. Citations in this document: §2.3, §2.3, §2.3, §2.3, §2.3, §2.3, §2.3, §2.4, §2.6, §3.2, §3.2, §4.

[63] Andy Greenberg, *Leaked NSA doc says it can collect and keep your encrypted data as long as it takes to crack it* (2013). URL: https://www.forbes.com/sites/andygreenberg/2013/06/20/leaked-nsa-doc-says-it-can-collect-and-keep-your-encrypted-data-as-long-as-it-takes-to-crack-it/. Citations in this document: §4.2.

[64] John Harrison, *The x25519 function for curve25519* (2024). URL: https://github.com/awslabs/s2n-bignum/blob/main/x86/proofs/curve25519_x25519.ml. Citations in this document: §4.4.

[65] Luke Henriques-Gomes, *Robodebt: court approves $1.8bn settlement for victims of government's 'shameful' failure* (2021). URL: https://www.theguardian.com/australia-news/2021/jun/11/robodebt-court-approves-18bn-settlement-for-victims-of-governments-shameful-failure. Citations in this document: §1.

[66] Theodore P. Hill, *How to publish counterexamples in 1 2 3 easy steps* (2009). URL: https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1077&context=rgp_rsr. Citations in this document: §2.3, §2.3.

[67] Joe Hurd, *Formal verification of probabilistic algorithms*, Ph.D. thesis, University of Cambridge, 2003. URL: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-566.pdf. Citations in this document: §4.4.

[68] Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, Bertram Poettering, *Cryptanalysis of OCB2: attacks on authenticity and confidentiality*, in Crypto 2019 [**43**] (2019), 3–31. URL: https://eprint.iacr.org/2019/311. DOI: 10.1007/978-3-030-26948-7_1. Citations in this document: §4.3.

[69] Jonathan Katz, *Publication of Koblitz's article questioned*, Notices of the AMS **54** (2007), 1454–1455. URL: https://www.ams.org/notices/200711/tx071101454p.pdf. Citations in this document: §3.2, §3.2.

[70] Jonathan Katz, Yehuda Lindell, *Introduction to modern cryptography*, 3rd edition, CRC Cryptography and Network Security Series, Chapman & Hall, 2021. ISBN 978-0815354369. Citations in this document: §1, §3.1, §3.1, §3.1, §3.1, §3.1, §3.1, §3.1, §3.1.

[71] Jonathan Katz, Yehuda Lindell, *Errata/typos for "Introduction to modern cryptography, third edition" (Last updated July 23, 2024)* (2024). URL: https://web.archive.org/web/20241218063041/https://www.cs.umd.edu/~jkatz/imc/errata-3rd.pdf. Citations in this document: §3.1.

[72] Joe Kilian (editor), *Advances in cryptology—CRYPTO 2001, 21st annual international cryptology conference, Santa Barbara, California, USA, August 19–23, 2001, proceedings*, 2139, Springer, 2001. ISBN 3-540-42456-3. DOI: 10.1007/3-540-44647-8. See [97].

[73] Neal Koblitz, *The uneasy relationship between mathematics and cryptography*, Notices of the AMS **54** (2007), 972–979. URL: https://www.ams.org/notices/200708/tx070800972p.pdf. Citations in this document: §3.3.

[74] Neal Koblitz, Alfred Menezes, *Another look at "provable security"*, Journal of Cryptology **20** (2007), 3–37. URL: https://www.math.uwaterloo.ca/~ajmeneze/anotherlook/ps.shtml. DOI: 10.1007/S00145-005-0432-Z. Citations in this document: §2.2, §2.6, §3.1, §3.3, §3.3, §3.3, §3.3, §3.3, §3.3, §4.2.

[75] Neal Koblitz, Alfred Menezes, *Critical perspectives on provable security: fifteen years of "another look" papers*, Advances in Mathematics of Communications **13** (2019), 517–558. URL: https://eprint.iacr.org/2019/1336. DOI: 10.3934/AMC.2019034. Citations in this document: §2.2, §2.2, §2.6.

[76] Leslie Lamport, *Some data on the frequency of errors in mathematics papers* (2022). URL: https://lamport.azurewebsites.net/pubs/statistics.pdf. Citations in this document: §2.4, §2.4, §2.4, §2.4, §2.4, §2.4, §2.4, §2.6, §3.2, §3.2, §4, §4.1, §4.3, §4.3.

[77] Pil Joong Lee, Ernest F. Brickell, *An observation on the security of McEliece's public-key cryptosystem*, in Eurocrypt 1988 [**58**] (1988), 275–280. DOI: 10.1007/3-540-45961-8_25. Citations in this document: §5.2, §5.2.

[78] Daniel Litt, *Reposting this from the other site at @ColinTheMathmo's request: I've recently been talking a bit about how difficult it is to carefully check even well-written mathematics. I want to try to explain something about this by telling the story of some errors in the literature that (in part) led to the two papers below. 1/n* (2024). URL: https://mathstodon.xyz/@littmath/113755426360011273. Citations in this document: §3.3.

[79] Frances Mao, *Robodebt: Illegal Australian welfare hunt drove people to despair* (2023). URL: https://www.bbc.com/news/world-australia-66130105. Citations in this document: §1.

[80] David A. McGrew, Scott R. Fluhrer, *The security of the extended codebook (XCB) mode of operation*, in SAC 2007 [**4**] (2007), 311–327. URL: https://eprint.iacr.org/2007/298. DOI: 10.1007/978-3-540-77360-3_20. Citations in this document: §1, §1, §1, §1, §1.

[81] Abdalrhman M. Mohamed, Juneyoung Lee, *Hoare Logic with the number of small steps explicitly annotated* (2024). URL: https://github.com/awslabs/s2n-bignum/blob/main/common/relational_n.ml. Citations in this document: §4.4.

[82] Douglas C. Montgomery, *Introduction to statistical quality control*, 8th edition, Wiley, 2019. ISBN 978-1-119-39930-8. Citations in this document: §4.1.

[83] Gen Nakamura, Gunther Uhlmann, *Global uniqueness for an inverse boundary problem arising in elasticity*, Inventiones Mathematicae **118** (1994), 457–474; see also newer version [**84**]. URL: https://eudml.org/doc/144244. DOI: 10.1007/BF01231541. Citations in this document: §3.3.

[84] Gen Nakamura, Gunther Uhlmann, *Erratum: Global uniqueness for an inverse boundary problem arising in elasticity*, Inventiones Mathematicae **152** (2003), 205–207; see also older version [**83**]. DOI: 10.1007/s00222-002-0276-1. Citations in this document: §3.3.

[85] Cathy O'Neil, *Weapons of math destruction: how big data increases inequality and threatens democracy*, Crown, 2016. ISBN 978-0553418811. Citations in this document: §1, §1.

[86] Christos M. Papadimitriou, *Computational complexity*, Addison-Wesley, 1994. ISBN 0201530821. MR 95f:68082. Citations in this document: §4.4, §4.4, §4.4.

[87] P. Jonathon Phillips, *A cross benchmark assessment of a deep convolutional neural network for face recognition*, in FG 2017 [**2**] (2017), 705–710. DOI: 10.1109/FG.2017.89. Citations in this document: §1.

[88] Leonid Reyzin, Douglas Stebila (editors), *Advances in cryptology—CRYPTO 2024—44th annual international cryptology conference, Santa Barbara, CA, USA, August 18–22, 2024, proceedings, part VI*, 14925, Springer, 2024. ISBN 978-3-031-68390-9. DOI: 10.1007/978-3-031-68391-6. See [33].

[89] Steven Rich, Barton Gellman, *NSA seeks to build quantum computer that could crack most types of encryption* (2014). URL: https://www.washingtonpost.com/world/national-security/nsa-seeks-to-build-quantum-computer-that-could-crack-most-types-of-encryption/2014/01/02/8fff297e-7195-11e3-8def-a33011492df2_story.html. Citations in this document: §4.2.

[90] Nishant Rodrigues, Brad Lackey, *Quantum lattice sieving* (2021). URL: https://arxiv.org/abs/2110.13352. Citations in this document: §1, §1, §1, §1, §1, §1.

[91] Phillip Rogaway (editor), *Advances in cryptology—CRYPTO 2011—31st annual cryptology conference, Santa Barbara, CA, USA, August 14–18, 2011, proceedings*, 6841, Springer, 2011. ISBN 978-3-642-22791-2. DOI: 10.1007/978-3-642-22792-9. See [36].

[92] Royal Commission into the Robodebt Scheme, *Report* (2023). URL: https://robodebt.royalcommission.gov.au/publications/report. Citations in this document: §1, §1, §1.

[93] Alfredo De Santis (editor), *Advances in cryptology—EUROCRYPT '94, workshop on the theory and application of cryptographic techniques, Perugia, Italy, May 9–12, 1994, proceedings*, 950, Springer, 1995. ISBN 3-540-60176-7. DOI: 10.1007/BFb0053418. See [10].

[94] Arnold Schönhage, Andreas F. W. Grotefeld, Ekkehart Vetter, *Fast algorithms: a multitape Turing machine implementation*, Bibliographisches Institut, 1994. ISBN 3-411-16891-9. MR 96c:68043. Citations in this document: §4.4.

[95] Adi Shamir, *Factoring numbers in $O(\log n)$ arithmetic steps* (1977). URL: https://dspace.mit.edu/bitstream/handle/1721.1/148919/MIT-LCS-TM-091.pdf. Citations in this document: §4.4.

[96] Claude E. Shannon, *Programming a computer for playing chess*, The Philosophical Magazine, VII Series **41** (1950), 256–275. Citations in this document: §1.

[97] Victor Shoup, *OAEP reconsidered*, in Crypto 2001 [**72**] (2001), 239–259. URL: https://eprint.iacr.org/2000/060. DOI: 10.1007/3-540-44647-8_15. Citations in this document: §2.2.

[98] Steven Skiena, *The algorithm design manual, third edition* (2020). ISBN 978-3-030-54255-9. DOI: 10.1007/978-3-030-54256-6. Citations in this document: §3.1, §3.1, §3.1.

[99] Nigel P. Smart (editor), *Advances in cryptology—EUROCRYPT 2008, 27th annual international conference on the theory and applications of cryptographic techniques, Istanbul, Turkey, April 13–17, 2008. proceedings*, 4965, Springer, 2008. ISBN 978-3-540-78966-6. DOI: 10.1007/978-3-540-78967-3. See [16].

[100] Jacques Stern, *A method for finding codewords of small weight*, in [**51**] (1989), 106–113. URL: 10.1007/BFb0019850. Citations in this document: §5.2.

[101] Vladimir Voevodsky, *Univalent foundations* (2014). URL: https://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/2014_IAS.pdf. Citations in this document: §2.2, §2.2, §2.6, §3.2, §4.1, §4.1.

[102] Peng Wang, Shuping Mao, Ruozhou Xu, Jiwu Jing, Yuewu Wang, *How to recover the full plaintext of XCB* (2024). URL: https://eprint.iacr.org/2024/1527. Citations in this document: §1.

[103] Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, Tal Malkin (editors), *Public Key Cryptography—PKC 2006—9th international conference on theory and practice in public-key cryptography, New York, NY, USA, April 24–26, 2006, proceedings*, Lecture Notes in Computer Science, 3958, Springer, 2006. ISBN 978-3-540-33851-2. DOI: 10.1007/11745853. See [15].