An Oral History Interview with
Daniel J. Bernstein
Conducted by Gerardo Con Diaz,
University of California, Davis
on 16 August 2024
via Video Conference

Charles Babbage Institute
Center for Computing, Information & Culture
University of Minnesota, Minneapolis

Abstract: This oral history interview is sponsored by NSF 2202484, "Mining a Usable Past: Perspectives, Paradoxes, and Possibilities with Security and Privacy," at the Charles Babbage Institute, University of Minnesota. The interview is with Daniel J. Bernstein, Professor of Computer Science at the University of Illinois at Chicago. Bernstein reflects on his early life in New York, his formative exposure to mathematics and computer science, and his long-standing interest in cryptographic security. He discusses his work in algorithm design, the development of cryptographic tools, and his legal challenge to U.S. export controls on encryption. The interview explores his views on academic freedom, adversarial design, and the relationship between cryptographic practice and public interest.

Keywords: cryptography, encryption export controls, algorithm design, academic freedom, secure communications

Con Diaz: It's August 16th, 2024, and this is an interview for the Charles Babbage Institute's National Science Foundation project, "Mining a Usable Past: Perspectives, Paradoxes, and Possibilities in Security and Privacy." I'm joined by Daniel J. Bernstein, professor of Computer Science at the University of Illinois at Chicago. We are conducting this interview through Zoom. Let's begin with some basic biographical questions. Can you tell me when and where you were born?

Bernstein: Well, I don't really have such clear memories of exactly what the event was like, but they tell me it was in New York in 1971. Not in New York City, in some hospital on Long Island.

Con Diaz: Can you tell me about your parents?

Bernstein: Yeah, they're both into technology and stuff and so from an early age, I had exposure to math and computer science and science generally. My mom for many years worked at the Protein Data Bank at Brookhaven National Lab. She was building up the main database of structures of proteins, crystallographic information. And my dad also had various jobs in math and computer science, mix of Brookhaven Lab, New York University. So again, various science, math related things. From very early I had exposure at home to—"yeah, this is fun stuff."

Con Diaz: Do you have any memories that stand out to you as key early moments of thinking about science or math with your parents?

Bernstein: So one story from them. I don't remember this particular one happening, but they said that I was very young, and we were in some store buying groceries and then my mom said she was looking at something that costs $2.49 and wondering how much two of them would cost. And I was coming out with the answer. I was too young to remember that. I mean, as far back as I do remember, yeah, always loved math and computers.

Con Diaz: Where did you attend elementary, middle, and high school?

Bernstein: Bellport, New York. That's where I grew up. Very small town, about 3000 people. And I guess we've heard about some even smaller towns out there. But that's where I grew up. And the high school was nearby. I mean, there were multiple levels of school. The smaller ones were actually in Bellport. The bigger one, Bellport High School, served multiple communities nearby and was a little distance away. But anyway, small schools, not at all urban environment.

Con Diaz: What was it like growing up in that small town?

Bernstein: When you're young, you know, you have people around you—your friends, your family. You don't realize until you've had some time living in cities how different it is. 3000 people, in some sense, it's actually a lot of people. There's quite a few people around and lots of people to get to know, and, you know, other kids to play with. And then the school itself, since it was serving more than just Bellport, there must have been hundreds of people in the graduating class. It wasn't like when I hear about really rural communities, then sometimes there's these stories about like 10 people in the class. It was definitely bigger than that.

Con Diaz: And you graduated high school at 15, is that correct?

Bernstein: Yeah, that's correct.

Con Diaz: How did you manage that?

Bernstein: Well, I was enjoying the schoolwork, particularly science-related stuff, but I was also doing fine in other classes. There were various meetings between my parents and the school to decide that I could skip some of the grades.

Con Diaz: In 1987 you placed fifth in the Westinghouse Science Talent Search. Do you recall any details about the algorithm that you developed for that contest?

Bernstein: What I wrote at that point was a little paper called *New Fast Algorithms for π and e*. This was to compute π to many digits or to compute to many digits. So, these mathematical constants where there's many algorithms for this and there's much bigger computations that people have carried out these days. But anyway, this was saving a little bit in some of the algorithms for computing lots of digits of pi.

Con Diaz: And let's talk about computing. Let's talk about your early experiences with computers and computation. How did you get your first computer? Do you remember?

Bernstein: I learned programming without having a computer at home. I remember this book on Basic. Now, Basic is a kind of controversial programming language. You can find, for instance, some quotes about how Basic causes brain damage from which programmers will never recover. There's actually a better version of Basic, like mainframe Basic, that was around before the sort of personal computer Basic that people were exposed to. I think this book on Basic was aimed at that. Although, honestly, for a kid, it doesn't matter. It's just like, here's how you can instruct a computer to do something. And it really doesn't matter what the language is. Anyway, I remember that book. *Conversational Basic*, I think that's what the book was called. It's probably possible to find that online or at least some records of where one

can dig up the book. Anyway, I'm reasonably sure that was my first programming book.

Now I didn't have a computer. What I did have was, well—parents would sometimes take me to Brookhaven Lab where there were a bunch of computers. Of course, I wasn't allowed to use any of those. The closest I got to the computers was like, there's these punch cards, which are little cards that have a rectangular array of spots where you can punch holes in them. And that used to be the main way to put programs into a computer, punching spots on these punch cards, and then feeding them into the computer where the computer would say "oh, this is an A, this is an X." My exposure to those punch cards was like, you could play with the punch card machine as a kid. So, you know, feed in the punch cards and make a little spaceship picture with the punching machine.

When I was actually programming, what I got to do for that was there was a friend who had access to a computer and he was able to take little programs that I wrote in Basic over to the computer, and like a week later would come back with the result. So, for him, long story of how he had access. So, I would write this program and later get results back. And that sort of environment of getting very slow responses when you're writing code certainly teaches you to be very, very careful in writing the code and not make mistakes—to think through what it's gonna do before you wait for the computer's response. Of course, with computers responding much, much faster, you don't have to worry about this so much. That was my very first exposure then.

It must have been 81 or maybe early 82 that my parents got an IBM PC. So, this was like the original Model 5150 IBM PC and suddenly we had a home computer. And I think also around that time, schools were starting to get the Apple II as a computer available in schools. I don't remember the order of when I was first seeing an Apple II in school versus the IBM PC at home. Then, some time after that, my parents got an Apple II for home as well. And then they were kind of surprised when I was going down to the computer room and they were somehow expecting I would like immediately come back and go, "Ooh, ooh, an Apple II, an Apple II!" And, well, I didn't, and they came into the room after a while wondering. I was sort of playing it cool. And they were laughing, going like "Okay, so apparently next we're gonna have some little baby computers from the IBM and the Apple and just keep expanding the computer family here." Anyway, so starting from around age 10 or 11 had some computers at home to play with.

Con Diaz: Could you tell me now about your time in college? I gathered that you started at Princeton and moved to NYU, is that correct?

Bernstein: Yeah, I transferred to NYU in 89 and then finished up at NYU in 91.

Con Diaz: Why did you choose to transfer?

Bernstein: NYU was a better fit for me. I mean, okay, one obvious thing, the urban environment. Princeton—there's lots of nice things to say about it academically. But socially, I'm definitely much more of a city boy at this point, even though that's not how I grew up. I was starting to understand my preferences at that point.

Con Diaz: And how did your interests start to gravitate towards number theory?

Bernstein: Good question. I always liked various different parts of math. I think it wasn't clear to me at the beginning, even when I started understanding that there were these different areas within math and different styles of doing math, different cultures, I think still it wasn't clear to me what was best for me. And even now, I like number theory, but also various other areas. And sometimes when I see people who are purely into number theory, they are, I think, often having a different mindset from me. Anyway, it is one of various things that I was interested in. And I think it was just coincidence that I was getting into various number theoretic things just earlier than some other things and happened to keep pursuing that direction.

But I don't think there's something in my mindset that's like, "ooh, that's clearly the right thing for me." One of the big splits people talk about in math is between algebra and analysis. Analysts make fun of algebraists: "All you do is say things are equal to each other. And if they're all the same, then, you know, they're equal. So what's so interesting about that?" Analysis, it's about things being close to each other. And somehow on that spectrum, I'm actually more on the analytic side than the algebraic side.

Mathematicians sometimes make jokes about how you eat corn cobs. There's some questionable studies which say that people who like algebra eat their corn cobs going in rows along the corn, which to me sounds totally crazy. I eat corn going around in circles around the cob. They say that analysts like their corn going around in circles. So that says that I'm an analyst, apparently.

Con Diaz: I want to pick up on a point you made, that you noticed that some number theorists tend to have a different mindset than the mindset that you had. Would you tell me about this mindset, about this difference that you noticed?

Bernstein: One example is, do you pursue abstraction for the sake of abstraction? So, this is in math and also in programming. There's value in having a general-purpose tool. And if you can apply something to more situations, of course, that's a good thing. And one of the ways that people apply things to more situations is have it more abstract where you need to sort of do a translation from your original situation into a framework which lets you apply this general-purpose tool that handles anything in that framework. But then that translation step of abstracting what you're actually doing to that more

general situation, so as to be able to apply a tool, for me that feels like a cost—having to say, "oh I'm rephrasing what I'm doing in this more abstract framework so as to be able to generalize."

The generalization is good and the abstraction, for me, it's not good. It's a cost that we pay in order to generalize. Whereas for a lot of people doing pure algebra and a large chunk of number theory, they'll say that abstraction is actually a good thing. Like, you wanna study abstraction for its own sake. And for me, this is weird. Abstraction is bad.

I think often people confuse this and they'll say abstraction is good because what they mean is that generalization is good. But sometimes people are saying, "No, no, these are different. And we like abstraction because that's a fun thing to do." For me, there's some language to learn, and it's interesting, but it's not what I enjoy about doing these things. And again, this is something where math and programming and lots of areas of computer science have this kind of tension. I find it particularly striking in some areas of math where people are really in favor of abstraction for its own sake.

Con Diaz: You got into hacking crypto in the early 1990s. Can you tell me more about how this interest was born?

Bernstein: There's this particular book that had an example of public key cryptography. I think that was the first time that I saw something about public key cryptography. It was specifically about a cryptosystem by Robert McEliece, a coding theorist who was using the mathematics of this in order to build a cryptosystem. There's this spy versus spy aspect of cryptography, which is interesting that people are aiming for different goals, there's the attacker trying to spy on your conversations, and there's the defender trying to stop the attacker from doing that. That's a sort of intrinsically interesting thing. And then the presence of the mathematics in this—that's something which I also really enjoyed.

I think another big burst of this that I remember from later, this must have been around 89, a particular bookstore at NYU. I don't remember the name of the bookstore, but I remember many hours standing in that bookstore reading a book by Neal Koblitz on number theory and cryptography. I did buy the book eventually, but I was just standing there and couldn't put it down. And I was like, wow. It was really fun stuff. That was a big source of interest. It wasn't as early as various things that got me interested in math. It's like 89. I was 18 or so.

I made a conscious decision, even though I was learning more and more about cryptography in like 89, 90, 91, and after that, I made a conscious decision to not be focusing on that as a topic. I was getting into grad school at some point. So, 91 I was off to grad school. I wanted to be in math and was also interested in computers. And so, I was naturally doing

computational math, computational number theory in particular. But I did not want that to be cryptography. And a major motivation for that was that, well, even though it's really exciting, it's obviously something which has legal issues.

I was already aware very early in studying cryptography that this is something where the government has constraints on it. It's not something that you can just casually be working on and publishing in. And then lots of the information that people have about it would be secret. And so, it's one thing to find something intellectually exciting. It's another thing to say, okay, is this actually, like, a safe topic to think about? And so, I decided, well, no, let me just stay away from this. Even though some things I was doing in grad school were maybe inspired by cryptography or related to it, they are not the same topic and don't give me this feeling of danger.

Con Diaz: Let's talk about Snuffle. What was Snuffle and why did you create it?

Bernstein: Well, let me explain one of the weird things about the law.

So, generally, part of the law made sense to me given the incentives of NSA, the National Security Agency. They wanted to have a monopoly over cryptography: NSA gets to do cryptography to protect their own communications, but nobody else gets to have it. That's basically the objective. Now, of course, you can't always get what you want. But the legal tool that they had was export laws. And they said any cryptographic tool that you try to export is something which needs to be below a certain security level, which then meant that they could break it. And anything above that security level, no, you're not allowed to export without an arms license.

This language of export and arms dealers and so on, it makes you think about guns and tanks and weaponry. But what they were actually talking about was publications. So, people who were writing papers—in the late seventies, academic cryptography was really starting to heat up with the discovery of public key cryptography, and there was an NSA employee, Joseph Meyer, who wrote a letter to IEEE about an important article on public key cryptography. And Meyer said, "Hey, what you're doing there is violating US export laws." And of course, for academics, it's kind of surprising to hear that stuff that you're thinking about—your job as an academic, you're reading stuff, thinking about stuff, writing stuff, and then you're not allowed to publish what you're writing because there's a law about it.

What's going on here? IEEE fought against that and published, but still this scared people. According to the law, NSA actually had a case there. They actually had a law which was saying "yeah, you're not allowed to publish anything about cryptography." And the way the law phrased that was, you're not allowed to export cryptography.

That's not what Congress actually said. They didn't say cryptography. They said, you're not allowed to export arms. But Congress is kind of lazy. So what does export mean? Well, they said somebody else will figure that out. So, they delegated to the executive branch to say what export means. And what do arms mean? Well, Congress is lazy. So, they said the executive branch will say what arms means.

So, then you have the executive branch saying, “well, okay, we have to define export, and we have to define arms.” Well, arms, I mean, some things are kind of obvious. Some things are less obvious. Like you have a ship that's going to sail for a while. It's gotta have all sorts of features that it can manage to sail. And some of those features include: You have to have really good paint on the outside of the ship. Well, if that paint is able to last for more than, I forget the number of months, if it's, like, really good quality paint, then it is arms according to the law, because, well, that's something you would only use for a warship. So really good paint is arms. You have the decision about what arms means delegated to people who are thinking not just, oh you know, what do arms mean to a normal person, but what can we get away with regulating? Then they start regulating paint because they feel like, “yeah, we don't want anybody else using really good paint on their ships.”

Part of this was delegated to NSA and NSA said, yes, cryptography is arms, and export includes not just sending something outside the US. As an example of the ambiguity, suppose that you're sending something into space, then is that export? I mean, it's sort of leaving the country, but, well, not really. I mean, you know, depending how far something goes, it's not actually leaving the country, or is it? Well, okay, the rule is if it comes down in another country, then it's export. That kind of makes sense. But somebody has to write this definition. And then the NSA contribution to this was saying that, yeah, if you have cryptographic information being published, even within the US, if any foreigner sees it, which of course any publication is gonna be seen by some foreigner, then that counts as export.

So, all publications basically count as export because there's gonna be some foreigner seeing it. So, when you translate the words into what does this actually mean, then publishing cryptography is export, even if you're just publishing within the US. And so, NSA was right that publishing things was illegal. Politically they wouldn't have been able to do anything about this. I mean, imagine the scandal if NSA actually took an academic publishing something and tried to throw them in jail for violating the law. Of course there's a First Amendment question there. And of course, NSA didn't do that, but they did go after people who were providing cryptographic software, for instance, Phil Zimmermann for creating PGP, Pretty Good Privacy. I think the way that he was distributing it, it's not even clear that he was doing anything that was getting it outside the US, but, well, even distributing things within the US, again, that's export. I don't think there's any evidence that he actually ever sent the PGP code to any foreigners, but,

well, they were harassing him for quite a few years, and he was certainly getting very close to violating the law, even if he didn't actually violate it.

So—distributing software, is that different from writing a paper and distributing the paper that says, “here's how you protect yourself by encrypting data”? The software says, “here's how you protect yourself by encrypting data.” It's in a different language that a computer can understand. Nowadays the computer can just understand the paper. But back then, well, there was like papers for humans to read, and software for humans and computers to read. Can the computer understand this? That's the software.

And, okay, according to NSA's perspective, all of this was, they didn't want people doing any of this. They didn't want any public explanation of how people can defend themselves against government surveillance. And so they were doing whatever they could get away with, trying to stop the public use of cryptography, except at a low security level. They made that exception because, well, that was giving industry a way to say, oh, yeah, we can do something that sounds like we're doing cryptography. And, I mean, it sounds like we're providing some security, it's meeting some market need. It's sort of this palliative for people who feel like there's some security threat, while being low enough security that NSA could still break it. And so that's what things were like at the beginning of the 90s.

Con Diaz: So you noticed all of these things happening and so you decided to write Snuffle because of these contradictions and ambiguities. Is that right?

Bernstein: Yeah. There was one particular contradiction that led to Snuffle. Politically you can see where the contradiction is coming from. Scientifically, it makes no sense at all. So, here's what the dividing line was for cryptography. They said, okay, you can export cryptography at a low security level, not at a high security level, but they also had an exception for authentication. So, authentication, for example, cryptographic hash functions are used to authenticate data, make sure that it's transmitted safely without hiding it, without making it confidential. You can have public information where somebody else can see that it's coming from you. Or you have two people who know each other and they wanna send data, which is not confidential, but they still wanna know that nobody's tampering with it.

Back then, NSA was apparently not doing so much tampering with data being sent through the internet. Nowadays it's known that that's something they're doing all the time, but, back then, it wasn't apparently on their radar screen. And, in any case, the law had an exception for cryptographic hash functions or other authentication mechanisms, these were allowed to be exported. So, you could publish software for a cryptographic hash function, for example. And, well, Ralph Merkle, I think he was at Xerox at that point, he designed some influential cryptographic functions, in particular a hash function called Snefru, named after a Pharaoh of ancient Egypt. And he was actually a little hesitant to publish the software for that, but John Gilmore

published the software for that, saying, look, there's an exception in the law for this, and you're allowed to publish this.

The thing is, scientifically what you're doing to build a cryptographic hash function versus what you're doing to build an encryption function to protect the confidentiality of data—it's really the same techniques. There are some slight differences, and the application feels very different. Like, are you protecting against sabotage? Are you protecting against espionage? Those are very different concepts, but the science of it—it's very much the same thing. And Ralph Merkle actually designed not just a hash function, also an encryption function. And there's lots of people who are doing both sides of that. So already I knew enough about cryptography in the early 90s to say, all right, you can just take one of these hash functions and use it to encrypt data. And it's just a few lines of software, a few lines to describe how it's done.

And well, that is Snuffle, which is not named after a Pharaoh of ancient Egypt. And Snuffle is taking Snefru, that hash function from Ralph Merkle, and it's using it to encrypt data. I wrote that. Again, just a few lines of code to do that conversion. It's much simpler to cross that line than to do something on either side of that line. And I thought this is making an interesting point about the dividing line. Like, even though politically you see where it's coming from, that NSA at that point wasn't caring about people who were protecting the integrity of data. But if you're allowing export of the techniques for that, it's a very, very small change from that to protecting the confidentiality of data.

And then I thought to myself, "well, okay, actually, politically, they are probably not gonna be happy with my saying this." So, I actually held off on posting the software for that. It was probably 1990 when I was doing the original thing, and then in 91 I was coming back to this and saying, "Okay, it's silly for me to be just sitting there in limbo. Let me find out whether in fact they think that I'm allowed to publish this."

And well, there's a procedure. The same procedure they use to say, is something low security cryptography or high security cryptography. That's some general procedure where, the way it's phrased in the law is, like, you can register as an arms dealer and say, "can I export something?" But, before that, you can say "wait, are we talking about arms in the first place? Or is this one of the exceptions?" And so, you can ask them for what they call the commodity jurisdiction procedure, managed by the Office of Defense Trade Controls. Is the following item something which counts as arms under your regulation? So, I wrote to them, and they're like, "oh, yeah, cryptography, this is something that NSA handles." So, I was talking with NSA about this, and NSA took a look at it, and they were like "yeah, this is not within our exceptions. It looks like you're encrypting something with a long key here. And you know, we could maybe provide you with some advice on how to do some lower security cryptography. Would you be

interested in that? And then you could probably publish that." But they were saying no for what I actually wanted to publish.

Con Diaz: Where would you have posted this software?

Bernstein: The main discussion group at that point was called sci.crypt. This was on USENET. There were lots of newsgroups, and a lot of 'em were science. So "sci" was science newsgroups, and one of them was about cryptography, and that was sci.crypt. And that one had all sorts of discussions, not just about the science of it, but also about the related policies. That's where I learned a lot about how there are legal constraints on things and what people have done, how people have gotten in trouble, and what the law says. But it was mostly about the science. And, certainly, people were discussing all sorts of cryptographic details there. So that would've been where I would've posted it. Also, I don't remember when I set up my first web page, sometime in the 90s. But it wasn't so easy to run a web page at that point.

Con Diaz: So was sci.crypt where you started hearing about these legal issues?

Bernstein: I think so, yeah. It's certainly something where people knew. In the seventies, it turned into this big issue of, like, NSA threatening a publisher, that's something which lots of people heard about, articles about that in big newspapers. And so it was definitely in the air that people knew there was a big dispute. And then the question of, all right, what exactly are you allowed to do? There were frequent discussions of that sort of thing. And so, yeah, sci.crypt was definitely a place where people were talking about the science and about the legal constraints.

Con Diaz: By 93 you had already connected with Shari Steele, the director of legal services at the Electronic Frontier Foundation. How did the EFF become involved in all of this?

Bernstein: Well, that's maybe a better question to ask them, but John Gilmore was involved in EFF from the very beginning. And he was also certainly tracking what was going on in cryptography. And I don't remember exactly who reached out to whom or what the communication patterns were. But it was a natural thing for them to be interested in. I think they were already from the beginning looking around, like, "Hey, these laws are obviously unconstitutional, obviously causing a problem for lots of people." And this was really dragging down security and really exposing a lot of people to attack.

For me, what motivates the work in this is not just, "oh, it's scientifically interesting," but it's something where people really do have their computers being broken into through all sorts of security problems. And one part of that is cryptographic problems. I don't mean to minimize all the other security problems, everything has to be dealt with, but part of that is, of

course, we need to have our data safely encrypted to stop people from doing bad things to the data. And so just protecting the users in the end was a major motivation for me. And obviously a major motivation for EFF.

Con Diaz: One thing I noticed from your story is a high level of political engagement and awareness, at least in this very specific issue. Is this a more general aspect of your life? Were you very politically engaged at the time, or was this really a cryptography issue that just drew you in?

Bernstein: I'm very much driven by the effects on the real world. It's certainly intellectually interesting doing all sorts of things which are maybe not relevant to the real world, but for me, a big driver is saying, "Hey, I can do something which maybe actually protects people." To me, that really adds a lot to this. For many years I've been interested in politics, generally what effect this has, but I try to limit the sort of things that I'm talking about to things where I feel like I know enough to say something, and I can maybe make a difference.

A random example of that is how I was helping out for a while with tracking osprey nests. If you're trying to see what effect pollution and other aspects of the climate have on nature—and possibly at some point on humans—then you have to collect data. Well, I happened to be doing that for an environmental group, writing software which was tracking and graphing, maintaining a database of where some birds were having their nests, and what the effects were from some pollution from Brookhaven Lab, which according to some water measurements were potentially pretty dangerous. And, okay, one small part of understanding the effects of that to monitor what you can. This was way back in school.

And somehow, I was in the position of simultaneously knowing enough about it and feeling like, "oh, I can do some data collection and some software, which is useful potentially for understanding what's going on." But of course there's millions of different political issues. To the extent that I actually feel like I know what's going on and I can do something about it, then, yeah, I'll happily comment about the politics. But that's of course kind of constraining. There's many things where I only know superficially what's happening.

Con Diaz: In 1995 you obtained a PhD in mathematics from UC Berkeley, working with Hendrik Lenstra. Your dissertation was titled *Detecting Perfect Powers in Essentially Linear Time and Other Studies in Computational Number Theory*. Would you tell me why you chose this topic and what your dissertation did?

Bernstein: When I got to Berkeley, there were no particular course requirements, but there were tons of courses available and there were seminars on various topics. I was going to various seminars and listening. The more

comprehensible ones were sometimes labeled as colloquia. And there was a particular talk, this must have been late 91 or early 92, from Hendrik Lenstra about some algorithms in algebraic number theory. I remember the topic. I remember the algorithm. He's a really good lecturer. It's something which subsequently got kind of renamed as factoring into coprimes in the context of algebraic number theory.

This was used in some fancier computations, but just to boil it down to the thing where I was feeling like, ah, that's a fun little algorithm: Suppose that you have some numbers you're trying to factor. This is something which is supposed to be really hard until a quantum computer comes along. But there are various algorithms that people try, to factor the numbers. Sometimes what you can do is, you have a bunch of numbers where they might share a factor. So, you have like 20 and 15, and they both have a 5 in common, and you can compute that really quickly with an algorithm going back to Euclid, where you take 20 and 15 and you repeatedly subtract one from the other until you're down to 5.

And then you say, aha, now 20 is 5 times 4, and 15 is 5 times 3. And now that 5 shared factor, well, that's partially factoring these numbers. And then the 3 and the 4: well, 4 isn't completely factored into primes because it's 2 times 2, but it's mostly factored into primes. The 3 and 4 and 5, you could sort of pretend that they're primes. It's not completely factoring, but just by taking greatest common divisors of all the different numbers that you see running around, this sort of partially factors things. And once you're done finding every common factor that you can see, then you can sort of pretend you're working with primes. That's good enough for a lot of algorithms. For breaking RSA, it's not good enough, but you can ask, what can you do with that?

And then this particular talk had fancier things about, in the context of other number fields, what does it mean to do these factorizations and so on. I was coming up after the talk and asking if anybody had actually implemented the algorithm. And Hendrik then pointed me to another algorithm, which was something that people had only thought about implementing so far, which was the general number field sieve. Now there's a preceding algorithm which had just been implemented called the special number field sieve. Both of these are factoring algorithms where you're asking what's the fastest way without a quantum computer to break RSA for any reasonable RSA key size that people use.

For the 90s, RSA 512 was the dominant cryptographic system on the internet. And then RSA 1024 for many years. Nowadays it's typically RSA 2048. For all these sizes, if you ask what's the fastest algorithm to do it, then currently it's variations of this number field sieve. And at the beginning, when the algorithm came out in the late 80s, early 90s, then people first saw how to factor some special numbers, and that's the special number field sieve. And then people realized how, with some extra effort, you could generalize that to handle more numbers. That's the general number field

sieve. But there's some extra work to make that general version work, and nobody had actually written software for that. And I was certainly very open about, yeah, I like computers, and I like programming. Hendrik asked if I could implement this, so I did—I mean, I wrote something that just sort of worked, didn't have a lot of optimizations in it, but it was certainly following the basic path of the algorithm.

Anyway, more discussions after that, and discussing lots of different algorithmic problems. This is an example of how close I was getting to studying cryptography without actually doing cryptography as a research topic. It was clearly relevant, but not giving me the feeling that I would get into any trouble.

You were asking about the thesis. Well, the most difficult part of the thesis was detecting perfect powers in essentially linear time. Previous papers said that you could do this in polynomial time: cubic time, even quadratic time. I was looking at that and saying, I can do this in essentially linear time, I think, but, okay, can I actually do that? I figured out how to handle most of the cases, but well, sometimes you have cases which are harder to handle. And actually, that's where most of the work was, occasional failure cases. I think these days I tend not to spend so much time on dealing with the occasional failure cases, if it's clear when you try computer experiments that, yeah, everything's fine. Do we really wanna spend a very large number of pages trying to prove that everything's fine when the experiments are good enough? But, okay, you do get some extra confidence if you've actually really mathematically proven something. And that's what I did. And that was the main chapter of my thesis.

There were some papers years later on "Mining your Ps and Qs" and "Ron was Wrong, Whit is Right" which is referring to RSA versus Diffie-Hellman. If you're having some sloppiness in how you use randomness in these cryptosystems, then RSA can fail in ways where DH tends not to fail. And that's something where the algorithms to exploit those randomness failures actually come down to the same kinds of techniques of partial factorization, which is what the main chapter of my thesis was getting at.

Con Diaz: So you were adjacent to crypto without actually doing crypto. Were you keeping an eye on developments in crypto while you were doing this? Or were you completely focused on the computational problems in front of you?

Bernstein: I was definitely looking at lots of different things. There's lots of things that people are writing about that are interesting to read. And then I do have to make a selection of what do I write papers about or, at that point, what am I writing a thesis about?

There's sometimes when people feel like they have to focus more on one thing to get into enough depth and think like, "don't tell me about anything else." And I've never been like that. I like having lots of things to think

about, even if some of them are things where I'm definitely not writing a paper about it, or maybe I'm writing some software, maybe just educating myself about something. I like having multiple things to look at at once. If I get stuck on one thing, I can move to another thing.

Con Diaz: Now, after you received your PhD, you became a research assistant professor at the University of Illinois at Chicago's Department of Math, Statistics and Computer Science. What took you to Illinois?

Bernstein: I was applying for jobs and was thinking at that point, okay, I could go to Silicon Valley and write video games. I had a little experience doing that. Other people I knew were writing video games for a living. I felt like, yeah, that looks like a fun thing to do. On the other hand, the academic thing also seems to be working out, so let me try applying for some academic jobs.

And then University of Illinois Chicago had a particular attraction for the department not just being a pure math department. So, what you see in the name was mathematics, statistics and computer science. That was within the College of Liberal Arts and Sciences. There was also a department of electrical engineering and computer science within the College of Engineering. So, there were actually two departments with computer science in the name, one more math focused, one less math focused, in liberal arts and sciences for the math one, engineering for the other one. That other one then split later into electrical and computer engineering and separately computer science, which is where I am now.

In any case, the interdisciplinary aspect to it was definitely an attraction. Oliver Atkin was doing computational number theory at UIC. And this was something where I think the department was thinking, okay, Oliver is getting towards retirement age, and it’s a strength of the department having computational number theory, and so they were looking to hire somebody in computational number theory. So, this feeling of, it's a good fit, and a department that wants to have me, and Chicago also is a nice place to live, so comfortable decision to make.

Con Diaz: At the University of Illinois at Chicago you transitioned from research assistant professor to a tenure track position and led two NSF grants on algorithmic problems in number theory. Do you recall what those kinds of problems that interest you early on as a professor were? Were they extensions of your dissertation? Were there new ones?

Bernstein: At least half of the things that I was proposing to work on were things inspired by factorization. Just understanding how secure RSA 1024 is, or RSA 2048. Would breaking RSA 2048 today be possible if we took all of the Bitcoin machines and oriented them towards breaking RSA? It's not actually easy to come up with an answer to that question. It relies on a lot of analysis. There's lots of things to figure out about security. And in later

years, I branched out into looking at many more topics. But there were definitely lots of factorization related topics right at the beginning.

And so that was a major focus for the initial proposals on algorithms in number theory. Later on, I was writing proposals on, for example, high speed cryptography. Around the turn of the century, the cryptographic laws got a lot more flexible because of political pressure and legal pressure and so on. By 2002 or so, I was feeling like, "yeah, I can comfortably write papers on cryptography." I might even have had a few in 2001 or so feeling like I'm not gonna get in any trouble for this. Even if I have software, I'm not gonna get in trouble for this. So that made me feel comfortable working directly on cryptography.

Con Diaz: You mentioned 2001. One of the key turning points, of course in the US history of privacy and security are the attacks of September 11th, 2001. Can you tell me about how you experienced that day and if, and how the, that day or the months after that affected your life or work?

Bernstein: I mean, shocking, obviously. And for a year after that, I felt like I wouldn't feel comfortable flying. And so, it turns out, for example, to get from Chicago to Toronto for a conference by train actually is possible. It's kind of slow, but it can be done. I was not considering conferences in Europe for a while. Anyway, obviously people varied in their reactions to this. For a lot of people, it was revenge and, yeah, definitely had a big effect. And, obviously, within the surveillance community, people were very fast at recognizing their opportunity to drastically expand the surveillance that they'd been carrying out for years before that.

There were already big reports before 9/11, about the ECHELON network, for example. This is a big surveillance network that the European Parliament was investigating by 2000. And already it was very large-scale surveillance. And then, well, if you're a spy agency and you're seeing what happens with 9/11, then you might have some questions of, oh, should we have predicted this? Did we predict this? Could we have done something about this? And you also have reactions of, how can we watch everything that's happening in the universe from now on increase our budget so as to put cameras in every single room in the world, and make sure that we're never surprised by anything again. So, yeah, it definitely had an impact.

Con Diaz: Thank you for sharing that. Let's get back to the lawsuit, because in 1995, while you were about to graduate from your PhD, the EFF announced that they would be launching a lawsuit filing a lawsuit at the Northern District of California. How did you feel about this decision?

Bernstein: Of course the NSA story is always about, you know, the terrorists. Child pornographers. Drug dealers. Money launderers, I think that one's kind of out of fashion, but that was definitely part of the narrative back in the 90s.

Obviously, there are some terrorists out there, but it's not like the cryptographic controls were stopping terrorism. It was much more an excuse for stopping everybody else, you know, all the innocent people out there, from having strong cryptography. And that's something where there's a clear public interest in having the constitutionality of these laws reviewed by a court. So, this lawsuit was filed in February 95, if I remember right. My PhD, I filed the thesis in May 95. So, I was kind of busy, but EFF was taking care of legal stuff.

Con Diaz: What was your relationship? There were several lawyers named in the suits. What was it like working with these lawyers?

Bernstein: It was good. Lawyers have lots of different specializations. Some of them will be showing up at trial and arguing things. Some of them will be writing briefs, the legal documents saying "Hey, look, here's what the court should do." And they file that with the court. There's people who are doing research, there's paralegals who are running around, the interns. There's lots of people who are helping out with figuring out what to do. And then there's what you see if there's a televised trial, you see the trial lawyers, and maybe occasionally there will be some of the document research people sitting there, but they're not the ones who are speaking. The ones who are speaking are the ones who are specialized in that. There's always many more people involved who are figuring out what to do.

Con Diaz: Who was your point of contact?

Bernstein: We had meetings involving several people. The people who were most involved from the beginning were Shari Steele and Lee Tien. And then EFF found McGlashan & Sarrail, and Cindy Cohn, who later went to work for EFF. There were various other people involved, but certainly from the beginning those were the main contacts.

Con Diaz: The software industry was quite interested in this battle, even if they were not particularly public about their support. Can you tell me more about your sense of industrial responses to this lawsuit?

Bernstein: I have no idea about the sort of policies that different companies might have adopted, but clearly, in talking to people, it was everybody versus NSA. To the extent that there were a few FBI comments on things, the FBI was supporting NSA, but the rest of the world was against what NSA was doing. Whether it was individuals or companies or other parts of the government, for example people at the Department of Commerce were like, "wait, what is NSA doing? And why do they say that's a good thing?" And again, NSA always comes back to the terrorists and child pornographers and so on, but realistically those bad actors with serious motivations to do something

really harmful, they will also figure out how to get their communications done in secret.

The cryptographic export laws were having much more of an impact on the random people out there who were not putting intense efforts into protecting their data. If I'm just a normal internet user, normal computer user, then I'll be doing what's default, what's normal, what's provided by software easily. And when that is not secure, then it allows mass surveillance on what everybody's doing. So that's what the real battle was about, even though, again, NSA kept portraying it as, "oh, terrorists, et cetera." I mean, yeah, of course, stop the terrorists, but that's really not what this law was accomplishing. Everybody else out there was feeling like what NSA is doing is not the right thing, again except maybe the FBI.

FBI is louder now against encryption. FBI measures their results by, like, how many criminals are they putting into jail? And they need to have enough evidence to put people into jail. They were maybe a little worried about encryption sometimes getting in the way of that. I also feel like I'm trying to stop all sorts of bad things, including some crimes. And if we can stop the crimes from happening in the first place because it's just not even possible to carry them out, then that's even better than catching them. But, of course, the FBI doesn't measure themselves by how many crimes don't happen.

Con Diaz: Now, in 2002, you decided to represent yourself. Did I get that right?

Bernstein: Yeah. So around 2000 the laws were liberalized a lot, as I was saying before. But there were still some fragments of the laws remaining. And I worry about risks, even if there are things which are not definitely gonna happen. Like if something has a 10%, 5% chance of happening, then for me it's like, "Hmmm. Think through the consequences of that and make sure it's not gonna be a problem." If people are saying, "okay, hey, we're actually allowed to distribute cryptographic software at this point, seems like things are okay," then, yeah, it makes sense that that most people wouldn't be worried. At that point, for me, I thought it was important to keep going with saying to the court, "Hey, look, here's the potential problems."

That had the advantage of the court saying "yeah, okay, there, there can be potential problems, and if there's problems in the future, come back." And so, I have kind of an open invitation from the court system. If the government ever does something bad in the future regarding cryptographic publication, then I can come back to the Northern District of California about that. I don't particularly expect a problem here. I mean, it does seem like the policies from 99-2000 are staying in place even after 9/11. It seems like there has not been pressure that's changed the regulations back to something restrictive.

There are definitely continuing arguments. NSA and FBI, maybe FBI now even more visible than NSA with saying, "encryption is evil. Encryption is

stopping us from prosecuting crimes." And, yeah, it does seem like, if you're trying to prove criminals did something, you run into encrypted data and yeah, it's gonna be a problem. Where I feel like the FBI is just confused is where they're claiming that, if we regulated encryption, then we could stop the criminals from using encryption. And that's just not how it works. They're criminals, they're doing what they want, they'll figure out how to get their data encrypted. And same for the terrorist argument and the child pornographers argument. Yeah, there's people doing bad things, and if somebody could wave a magic wand and stop the evidence from being hidden, then that sounds like a good thing. But we don't have that magic wand. What we can do is make easy to use cryptographic tools for everybody else, normal people who aren't putting massive effort into this. Anyway, so the court proceedings after around 2000 were for dealing with potential future possibilities where thankfully things have not been a problem since then. And hopefully that continues to be the case.

Con Diaz: In 2002, the court also dismissed the case on ripeness grounds. Did you celebrate? What happened after the saga was done?

Bernstein: There's always this concern of what's gonna happen. But you know, also at some point I have to say, "okay, this is as good as the situation's gonna get." Ripeness is how a court says there's not a problem yet. There may be a problem in the future, but there's not a problem at this point, so we don't have to do anything. And having that on the record, that feels like, yeah, okay, there's clear procedures. Laws, regulations change all the time. And if there's something which happens in the future, then it's clear what the court procedures will be after that. Other people obviously also can go to court.

Con Diaz: And of course, the court also ruled that software was protected speech. Would you tell me more about this connection between coding and speech as you saw it at the time?

Bernstein: I filed a declaration for the court case saying how to compute the date of Easter. You're given a year, and you divide year by a hundred, giving century, and so on. What I just said and all the rest of it is actually something computer comprehensible even in the 90s. This was a program written in COBOL, the Common Business Oriented Language. That's another language that has a bad reputation, like BASIC, but, whatever, it's a perfectly readable programming language as long as you're not trying to do anything complicated in it. And then you can say things like divide year by a hundred, giving century, and that is a COBOL command. And exactly that comprehensibility is why COBOL, I think it was created in the fifties, the reason it was created was exactly to make something which is kind of verbose compared to most programming languages, but it's also very readable and people can understand it without any training really.

And so, I filed this declaration, here's this software to compute the date of Easter. And it's something that, as a human, you can just read through, try out. I remember Lee at one point was saying that he tried it out on whichever year and it worked—got the right date of Easter.

The point is, people who don't program will think software is, I don't know, some sort of weird thing that computers are somehow learning to do something, like in the Matrix, you know, get a little thing uploaded into your head and suddenly you can fly a helicopter. But when you see actual examples, especially readable examples, then you see that software is just instructions to do something. And the reason we call it software instead of instructions is simply saying that a computer understands it.

Now, one of the interesting consequences of AI is people realize something which the judges were actually emphasizing back in the 90s: computers are gonna be able to understand our normal speech, normal instructions people are giving to each other. And is that going to kill the First Amendment? Are we no longer allowed to have freedom of speech because computers can understand it? No, that's ridiculous. It's not what the First Amendment says. And so, software is just instructions, and, in the case of cryptography, it's instructions for how to encrypt your data or how to sign your data, authenticate, do hash computations, whatever it is. And so I was filing this and saying, “look, this is an example of software.”

In the district court, Judge Marilyn Hall Patel said the fact that instructions are something which is comprehensible to a computer doesn't take it out of the First Amendment. I think this is something where, for people who don't know what software is, then they will ask, “what does that have to do with freedom of speech?” And then when people know what software is and see examples of it, then it's like, yeah, of course. And, in the age of AI, I think it's even clearer that you can't say that just because a computer understands something it's no longer freedom of speech.

So that was one of the early decisions. The government was filing a motion to dismiss the case, saying there's no First Amendment issue, no constitutional issue. The government basically always does that. If they're sued for doing something unconstitutional, they'll say, “There's no constitutional issue here. Of course we're allowed to do this,” and what the judge said was “No, no, no. There is a constitutional issue here.” The core of the government's argument, I think more broadly, is that the government will say, “We're not regulating that because of its speech content. We're regulating that because of its effect.” Like, it has this effect on the listener, its effect on the community. That's the impact, the actions that we're trying to deal with. But they're still not allowed to regulate speech. And then the government in this case was saying it's not even speech to begin with, it's software. And then that's where the judge said, “No, no, no, just because it's software doesn't take it outside the First Amendment.” So that was the software is speech decision.

Con Diaz: And this is the first time a court had ruled that in the US right?

Bernstein: I think so, yeah. And also, I think the court's discussion of it was something that other judges looked at and found convincing.

Con Diaz: Apple cited it.

Bernstein: Courts can change their minds about things and even constitutions can change every now and then. I think the last amendment to the constitution was, like, congressmen are not allowed to increase their own salaries. They have to wait until after an election. But I think the speech principles here are reasonably well understood. And yeah, I think software as speech was first articulated in that decision.

Con Diaz: Yes. Okay, so now we're gonna move a little bit more to your research. From 2000 to 2005, you held an NSF career grant on computational number theory, cryptography, and computer security. Now, all this time, you've been connecting the dots for me about being a number theorist who is holding hands with cryptography, but now you have this grant that actually merges things and puts that merger first. Would you tell me more about this grant and about the kind of work that you transition once you allowed all these fields to come together?

Bernstein: Not just that grant, but also subsequent grants. In general, I feel like there's a big problem with computer security to start with. There's so many ways that people break into systems and there's more vulnerabilities that we know about, which maybe haven't been exploited, but it's a real disaster area. I think, with ransomware these days, people are seeing very much how exposed the computer systems are. But, even before that, it's something where anybody investigating computer security sees how weak the systems are, how many vulnerabilities there are, and, well, it's a big problem. And I don't want to exaggerate the impact of any particular approach to solving the problem, but let me say something that I like to do.

I've noticed that, in a lot of areas where people are making progress on things, what really promotes the progress is being able to measure how close are we to success. In medical research, people might say things like, here's some disease and are we able to cure it in 90%, 99% of the cases? Are we in cases where we can't cure it? Can we delay the effect for 5 years, 15 years? And so, people have these numbers that they use. And then, in computing, there's lots of things where people are saying, “okay, here's some software. I wanna make that software faster.” You're trying to make the computer respond faster and you feel like, there's this measurement which is driving progress.

But then there's some things where it's actually hard to measure and you see people flailing around and saying, “oh, well, we'll do something, but we're not sure whether it's working because it's hard to see whether it's working.”

And that's something where, around 2000 or so, I was starting to see this is a big problem. I think people had pointed this out before for computer security generally. And I was also seeing that it's a problem for cryptography, the difficulty of measuring what it is that we're doing. And, well, I think the question of how to measure progress towards security, it's a big topic. What I've been doing for research is saying, "okay, look, here's some measurements which seem good, they seem correlated with progress made so far. And let's just say these are good measurements and try to make progress within those." And this is something where already there's lots of interesting stuff to do, even if maybe it's not the perfect measurements, but I just feel much better having those there.

And then in the case of computer security, generally what I like is a very traditional notion of how we're measuring progress, which is the size of the TCB. So, the size of the trusted computing base you have—within your computer, you've got tens of millions of lines of code. A browser by itself is a massive number of lines of code. There's the operating system kernel, all sorts of other programs. Tons and tons of code. And the situation right now is that basically all of that code is able to take over your computer. When your browser is going to a website, and then the website has some programs, it's running on your computer to do all sorts of displays of things. Then, okay, those are programs which are supposed to be in a little sandbox that hopefully can't affect anything else you're doing, but that's kind of a permeable sandbox and you gotta be really careful to constrain those programs.

And what's doing the constraints on those programs? Well, it's tens of millions of lines of code which are saying, here's what your browser is allowed to do, here's what your operating system can do. And, well, okay, the concept of the TCB size is how many lines of code on your computer are able to violate your security policy? For instance, take over your computer or see your confidential data, whatever it is that you're trying, whatever your security goals are, which lines of code do you have to check to make sure that those can't violate your security goals?

If your system is structured in a way that says these millions of lines of code over here are not able to affect your computer because you've got a solid sandbox around them—I mean, sandbox terminology maybe suggests it's softer than what you actually want, which is you want that to be really jailed, it's really constrained, so that software cannot do anything bad—then, well, you still have to check the software that's doing those constraints and you have to check everything that's outside of this jail, but it reduces the amount of code where things can go wrong. And it feels appealing: the less code we have to worry about, the better off we are.

So, then what I was doing in that proposal, I was articulating various concrete ways to reduce the amount of code that actually can affect our system security. And this is something where, again, it's an old concept in computer security, goes back to the eighties. But it's not what most security

research does. And that concerns me because I feel like a lot of the things people do in security, they're sort of chasing after particular failures and saying, “we can recognize this failure as follows, we can stop this failure as follows”, but, at the same time, not addressing the problem of more and more millions of lines of code being added to the computer system. How do we address failures in that code? And then sometimes people are saying, all right, structurally, here's ways that we can actually make sure that we're not having a bigger and bigger problem, not fighting against the tide here. How do we redirect the tide? And so that's something that I like seeing in security and that I like working on in security. And then there's also cryptographic aspects to that. But that's the general principle.

Con Diaz: Now, can you tell me more about Salsa20?

Bernstein: As context, going back to my court case, there was this general idea of “okay, here's a simple way to encrypt something using a hash function where the hash function has all sorts of complicated things inside it.” As I learned more and more about cryptography and caught up on what people are doing and studied thousands of papers, I started saying, “okay, here's simpler ways to do those things inside a hash function.” Of course, you wanna put that inside encryption to protect confidentiality, not just integrity of things. And then more and more simplification got to the design of Salsa20.

There was an academic competition called the eSTREAM competition, which was run by a big consortium of European organizations. The consortium was run by academics, the committee looking at proposals had academics and industry people, and there were some companies involved in the consortium. Anyway, they were calling for proposals of stream ciphers, so encryption mechanisms which are able to handle large amounts of data efficiently. I was designing this thing and really trying to boil down what you have to do for encryption to the simplest possible thing, and at the same time was having these thoughts of, “okay, how do we make sure that that things are really secure? We wanna be able to see how close we are to security. Where are the threats? Can we measure that?” And so, I used what's called a round-based design which some ciphers do, some ciphers don't. A lot of stream ciphers don't do this. But I made sure to use that because that's something that gives us a measurement mechanism for attacks. So, I had this simple, reviewable from a security perspective, and also fast mechanism of encrypting data, and that's Salsa20, and submitted it to that competition. Did well there.

Con Diaz: Tell me what are you working on now and what has inspired you to approach these topics?

Bernstein: I'm always again trying to bounce between different things. But the main motivation at the moment is that we've got a big security problem right now with future quantum computers.

The issue is that we've got all these ciphertexts being encrypted with various public key cryptosystems like RSA and elliptic curve crypto. And those cryptosystems will get broken by a big quantum computer as soon as somebody builds a big quantum computer.

So, here's the attack strategy. Number one, somebody records data. Right now, your confidential data, you've encrypted it using RSA or ECC, somebody records it—somebody being, well, whichever large-scale attackers you're worried about, they are spying on your data, and they've been trying to record all of it for many, many years. And there's already, 10 years ago, there were news articles from some leaks from what NSA is doing, saying if they encounter encrypted data they store it forever to try to be able to crack it. So, okay, that's the attack. They're recording your data right now. You've encrypted your data, they're recording it, and when they have better computers later, they go back and they break your ciphertext, they figure out what the secret data was. So that's attack strategy number one.

And a quantum computer plugs into that, if at some point they have a quantum computer, let's say five years from now, 10 years from now. I've got bets about this publicly happening by 2032. I think that we're not gonna know when it happens secretly, when a large-scale attacker has it, but it could be years earlier because they've been investing in it for a long time, and they're probably ahead of the public somewhat.

But, okay, publicly, we're gonna see big quantum computers coming. At least it looks like that. Maybe something will go wrong, but the technological progress looks like early 2030s, maybe sooner, maybe a bit later, but not much beyond that. There's gonna be big quantum computers, which we know are able to retroactively break this data as long as they had the common sense to record the encrypted data.

There's also attack strategy number two, which is relying on human laziness, where we just wait on deploying post quantum crypto, meaning crypto designed to protect against the threat of quantum computers. If we wait on deploying this, then we see, let's say early 2030s, we see somebody publicly says, “oh yeah, here's how you can break these systems with a quantum computer. We've actually done it.” And then suddenly we're going, “oh no, now we have to protect ourselves.” And we try rolling something out, but it takes years and years and years to get something rolled out. Some things can get upgraded quickly. Lots of things cannot. Takes many years to get upgraded. And so, we're gonna have years of attackers with quantum computers who are breaking all sorts of systems, not just confidentiality, but also integrity, just taking over systems. Until we've got the systems upgraded, we're in big trouble at that point. That's attack strategy number two.

So, we have a problem right now that we are encrypting data in a way which is not secure against those future attacks. And if that data is something which needs to be secure, like medical information or legal information, all sorts of privileged information and things that need to stay confidential for many years, then we have to do something about that now. Otherwise, we're not protecting that data. We're only protecting it against the next few years and not long term.

Post quantum crypto. This is a term that I introduced in 2003 for at least trying to protect against this. Some systems which were identified as maybe protecting against this had already come earlier, but without a name for the general concept of trying to protect against quantum computers. Anyway, that's been the main focus of my current research.

Con Diaz: And if someone wanted to become more familiar with this term as you introduced it, post quantum crypto, what should they read?

Bernstein: There's various sources. Right now, just this week, the National Institute of Standards and Technology issued some standards for post quantum crypto. And one way to get started is just look at those standards, look at the news about that. And the standards have some reference lists, which is maybe not the easiest introduction, but there's another source, pqcrypto.org, this is a website that some of us run, which is tracking a lot of what's happening, pointing to conferences and some papers. There's so many papers on the topic, it's hard to keep this up to date. But that's one source. There's various introductory video series.

There's been some deployment so far. I'm optimistic that it will turn into the standard cryptography that everybody's using. It's just been standardized. The biggest deployment so far is in SSH, which is the standard tool for remote administration of systems. People who use SSH starting with version 9 of OpenSSH, they are using post quantum crypto by default. That feels good. There's some use in browsers which also feels good. Hopefully these systems hold up. I mean, at least we're trying to protect against future quantum computers. But there's no guarantees. There's a lot to do. Hopefully we'll manage to get it done.

Con Diaz: That's wonderful. Dan, thank you so much for joining me today. This was wonderful.

Bernstein: Thanks for your time.

Con Diaz: Bye-Bye Dan. Thank you.