Oral History Interview with

Steven M. Bellovin

June 14th, 2023

Video Conference

Conducted by Jeffrey Yost

University of Minnesota

Abstract: This oral history interview is sponsored by and a part of NSF 2202484 "Mining a Useable Past: Perspectives, Paradoxes, and Possibilities with Security and Privacy," at the Charles Babbage Institute, University of Minnesota. It is an interview with Percy K. and Vida L. W. Hudson Professor of Computer Science, and affiliate faculty member of the Law School Steven M. Bellovin, Columbia University, a pioneer and expert in security as well as a leading scholar in technology and law. The interview starts with Bellovin's recollections of his pre-college and college education and how he encountered programming. He then discusses his graduate education, his mentors and professors—Fred Brooks, David Parnas, and Brian Kernighan—and their influences on him. Bellovin briefly describes and summarizes contexts to his dissertation in formal methods, "Verifiable Correct Code Generation Using Predicate Transformers." Then the interview shifts to focus on USENET during his graduate school days and beyond. This includes sharing his thoughts on the personal computer revolution, democratizing computing, important concepts growing out of USENET such as "flame," "sock puppet," "trolling," "spam," "FAQ." Bellovin offers context to his joining AT&T Labs and his professional focus on computer security research. He shares the context of Morris Worm, as well as the origin of cryptographic authentication, the idea of firewalls, and his work serving on the Internet Engineering Task Force or the IETF. He discusses joining Columbia University and his research and teaching. The latter part of the interview focuses on his growing focus on technology and law.

Keywords: Computer security, privacy, USENET, Internet Engineering Task Force (IETF), cryptography, law, public policy, computer science, firewalls, internet security, technology and law, privacy, Morris Worm, cryptographic authentication, AT&T Labs, Columbia University, access control systems.

Yost: My name is Jeffrey Yost, and I'm at the University of Minnesota. I'm here today on a video conference, for an oral history with Percy K. and Vida L. W. Hudson Professor of Computer Science, Columbia University Steve Bellovin, who is also an affiliate faculty member at the Columbia School of Law. It is June 14th, 2023. And this is an oral history of the Charles Babbage Institute for Computing Information and Culture. It is being sponsored by a grant from the National Science Foundation, "Mining a Useable Past: Perspectives, Paradoxes, and Possibilities with Security and Privacy. So, I'll just begin with a few biographical questions. I understand you were born in New York City. Can you tell me what year you were born, and did you grow up in Brooklyn?

Bellovin: Yes, I was born in 1951. I grew up in Brooklyn, went to Stuyvesant High School, which today you would call a magnet school, with examination entrance for students interested in math and science. And for my undergraduate career, I went to Columbia University.

Yost: So, pre-college, were math and science your primary interests?

Bellovin: Absolutely. I knew early, no later than fourth grade, I wanted to do something like that. Going to Stuyvesant, once I got in, it was absolutely a no-brainer for me.

Yost: And, entering Columbia University, did you have an idea of what you wanted to major in?

Bellovin: Yes and no. I had actually learned to program. Had a fair amount of experience programming computers in high school, which back then, was a very unusual thing. That's a whole separate story but the school got a computer by the start of my junior year. I already knew how to program. So, the teacher in charge let the students who knew how to program run the thing. I started college as a math major because there was no such thing as a Computer Science major at the time. I ended up graduating as a Computer Science major, even though it still didn't exist there. I made up my own major, which I persuaded the Committee on Instruction to accept. And my part-time jobs all through college were computer programming part-time jobs.

Yost: So, in making up this major, what courses were there related to computing and software, even though there wasn't an official degree program in computer science? So, in short, can you expand a little bit on the courses and study that you did?

Bellovin: Sure. So, Math majors at the time were required to take, I forget how many credits of math and 12 credits of science. And I only needed six more credits of math to graduate. I just didn't feel like doing that. A friend had already managed this petition to the Committee on Instruction, so I tried it too. And for those 6 credits of math and 12 credits of science I had already taken, I substituted computer courses from the engineering school and the graduate school. There were some hardware-oriented courses on switching theory and a computer implementation course in assembler language, and a course on data structures. I don't recall what else I took. I was also doing an independent research project with some friends on one of the early computer chess programs. I don't recall if I got college credits for that or not.

Yost: Were there any faculty that stand out to you as being especially influential to you at that time in your life?

Bellovin: One professor who was almost more of a negative influence. He taught data structures, which at the time was considered by the course number an advanced graduate class. These days at Columbia, it's about a third semester undergraduate course for CS majors. And he didn't want me in the class, even though he knew me and knew I was qualified. "Why don't you want me in the class?" "Because this is a graduate class." "Don't you think I can do it?" "I know you can, but it's a graduate class. You don't belong here." I rather resented that and vowed never to do that to my students, where I am for a faculty member, and I have stuck with that. I have never once kept a student out of my classes because they don't qualify as graduate students. The professors who were really important to my professional development were in graduate school.

Yost: And did you work at all in between your undergrad and graduate school, or did you go straight to grad school?

Bellovin: I went straight to graduate school. That's one of the oddities of my career. Even in retrospect, I have no idea why I went to graduate

school. It just seemed like the next thing to do. After high school, you go to college; after college, you go to graduate school. And why I should have concluded this, I don't know, since neither of my parents had graduate degrees. It was only years later that I learned my father had enrolled in graduate school, but never finished. And my mother took graduate courses, never intending to get a degree, but that probably didn't start until after I graduated.

I have no idea why I went. I was extremely qualified to get a very good job as a programmer. At that point, I had six, seven years' experience programming, including a lot of operating system internals. So, I could have gotten a good job, but it just seemed like graduate school was the right thing to do. And even in retrospect, I don't know why. But again, I had worked for four years doing systems administration, systems programming in college, and was doing system administration in high school for two years. So, I had a fair amount of work experience before starting graduate school.

Yost: Was this programming you did for the university? Was it for corporations or a combination?

Bellovin: There was a combination. My first year in college, I worked at one of the schools, Teachers College which had a very large IBM 1130 system. The next year I went to work for IBM—the same machine, IBM 1130. They at the time had a small building across the street from the Columbia campus. IBM shut that facility down and moved everyone up to Yorktown Heights. For my remaining two years in college, I worked a mile uptown at City College of New York, which was then the computing hub for the entire city university system. And we had remote terminals and so on that we served.

Yost: And can you speak a little bit about the computing center at Columbia at the time you were there?

Bellovin: So, this computing center was fairly large for its day. It had a supercomputer class mainframe in IBM 360/91. And the attached support processor, a 360 model 75 was what in other places would be considered a fairly large computer. And here it was just handling support I/O [input/output] and so on for the 360/91, which as I said, was a supercomputer class machine. The ironic

thing, of course, being a supercomputer, it had a grand total of two megabytes of storage, which in those days was a lot.

Bellovin: Couple of banks of tape drives, couple of banks of disk drives. disk packs were either 30, about 12 megabytes or 27 megabytes. So yes, small by today's standard. I think my wristwatch has got more computing power and *[laughter]* certainly more storage space. But yes, it was serving the university and the research community up to and including having a bubble chamber for the physicists inside the machine room.

It was mostly punch card, though there was an interactive text editing remote job submission system that they had. Originally one called CRJE, Columbia Remote Job Entry System. Later they got some software called Wylbur, same sort of thing. Edit files, submit the jobs, retrieve the output, but primarily a punch card system. And you would get the output back an hour or a few hours later, depending on the resources required.

Yost: So, it was exclusively batch back then. There wasn't a time-sharing system.

Bellovin: It was exclusively batch. There was no time-sharing. I had a bit of time-sharing experience when I was in high school. During one of the summers, I gained access to a terminal for the Dartmouth Time-Sharing System. So, I taught myself BASIC and ALGOL 60. At Columbia, I used PL/I, APL, FORTRAN and IBM Assembler Language, mainframe 360 assembly language.

Yost: And in deciding to go to graduate school in Computer Science, can you take me through the process of schools you considered and were you aware of David Parnas's work in deciding on UNC?

Bellovin: Parnas didn't come to UNC until after I had been there for several years. So that wasn't it. One of my friends had applied to Chapel Hill. I didn't really have a very good or structured process for deciding. One of my friends applied to Chapel Hill though he went elsewhere. I don't recall where, but it seemed like a good place. I applied to MIT and did not get in, which was really nice because they had rejected me for college also, so I thought that was a fitting event.

But Chapel Hill was actually about the best spot for me. It was a small department where Fred Brooks, the department chair, was a tremendous influence on my professional career. And I could speak volumes just about Brooks. He was the department founder, department chair, and I took four courses from him, which knowing what I now know about academic lifestyle is utterly insane. But I took two lecture courses and two seminar courses from him in my first two or three years there.

Yost: Are there certain things that you feel that helped shape you and your career, that you took from Brooks?

Bellovin: There were a lot of things. Brooks is very famous as the person who managed the design of the [IBM System] 360 series and managed OS [operating system] 360, and then taught a course on software engineering, where basically the theme was why was software so much harder than hardware? The manuscript for *The Mythical Man-Month*, his most famous book, was our text for the class.

So, I learned a lot about what it took to manage software projects, why they were hard, what you should do, what you shouldn't do, and so on. He was also a brilliant teacher. One of the best teachers I have ever had from classroom teachers, from a purely instructional basis. He knew how to motivate students. He knew everyone, knew everyone very well, their strengths, weaknesses, and so on. And you couldn't get too much of his time because he was so busy, but if you needed to talk to him, he would talk to you. He was my advisor, my first failed attempt at a dissertation. And it was my thesis proposal stage. He and the rest of the committee warned me that it was a high-risk project, which I knew. Had to do with automatically building code generators for compilers.

And I said, well, I'm young. I'm single with no family responsibilities. It's a high payoff project. Let me try it. And after a year, and somewhat retargeting it because of what I had found in my literature search, I realized that my approach was equivalent to the halting problem. I decided that this was not a good thing to try to solve for my dissertation. And decided I was going to take a year off from school and then go back and take a master's degree, and stop. And I actually had a faculty offer as a lecturer elsewhere. Back in this mid '70s, you could get a faculty position in Computer

Science, especially in a small school with only a masters'. There weren't a whole lot of CS PhDs around.

Yost: Right.

Bellovin: But Brooks was of the opinion that if I left, I was not going to come back and finish. He was quite possibly correct. And he wanted me to finish. He had a great deal of confidence in me and knew me very well, maybe better than I knew myself. So, he fought the university bureaucracy to whatever extent was necessary. He never told me, and made me a faculty member in that department, in my own department for a year. And I then went back to being a PhD student, my ultimate dissertation was with Dave Parnas who had arrived a year or two earlier.

At the time I thought I was doing programming languages. My dissertation ended up being formal methods because I was a systems person, but I decided to get some more breadth by doing a formal methods dissertation. Theory being my second strongest subject in graduate school. Yes, that's an example of what I mean though, about Brooks taking care of people. He wanted me to get my doctorate. I was one of his protégés. Even when I wasn't his PhD student, he was going to make sure I had every opportunity to get my doctorate.

I'll give you another example of something he did. When I was teaching in that department at that time, PhD students had to teach one semester of introductory programming and I did it four times because I loved teaching. You had full classroom control. You were not a TA. This was full classroom control. And I had one student who started slowly. Didn't get it, and suddenly something clicked, and the student finished very, very strong. And when I looked at the final averages, he was a hair below the cutoff between, I think it was B plus and A minus. But because he had finished so strong, I said, look, he really knows this material. I'm going to give him the A minus, jumping him over the student who had a B- who had a slightly higher-class average.

That student filed the grade appeal against me saying that this student with a lower average is going to get an A minus. He deserved it too. And Brooks fought hard up to the dean's office saying, "You can't overrule a faculty member on a matter of

grading." “This is his evaluation. You can’t do it." He lost, but he fought very hard for me. When he trusts someone, he trusts them unreservedly. And it’s something I have tried doing teaching. When my TA’s assigned a grade, unless it’s grossly unreasonable, I’m going to back the TA. I have given them that responsibility.

There’s another anecdote that will show that. Towards the end of my time in graduate school, we got a couple of [Digital Equipment Corporation, or DEC] VAX-11/780s. By that point, I was no longer one of the official system administrators. I was supposed to be just finishing my dissertation, but I had the root password. There was a faculty member who didn’t really understand the difference between real memory and virtual memory and fired up eight simultaneous jobs, each of which over committed the real memory of the system. You try to run eight in parallel, the system’s going to thrash and no one’s going to get any work done, including that professor.

So, I logged on, used the root password, and stopped and killed seven of the eight jobs, letting one of them continue, emailed the professor, told them what I had done, told them what he would have to do to resume them, “one at a time, please.” And then emailed Brooks’ department chair saying, I know this isn’t my job anymore, but it was necessary. And Brook’s answer was, “If you have the root password, you have not only the authority, but the responsibility to use it that way.” If I was going to be trusted with the root password, I had to use it.

Yost: Oh, absolutely.

Bellovin: In a situation like that. But again, that’s the kind of person he was.

Yost: So, your dissertation was in formal methods and entitled, “Verifiable Correct Code Generation Using Predicate Transformers.” Can you tell me how you came to this research topic, and briefly describe the dissertation?

Bellovin: Okay. So, at the time, I was most interested in programming languages and compilers. I had the feeling, and that matter still do to a fair extent, that what you can say in a programming language will affect how good the program is. These days, I had focused on the security of course, but correctness was certainly a very

important thing. So, as mostly a programming language person, I was looking for a way to combine them.

Dijkstra had just written a book called *A Discipline of Programming*. Parnas loved that book. And one of the things in the book was a particular programming language that Dijkstra had defined. And he also gave the formal semantics of the language using a technique called predicate transformers, which Parnas also liked. So, it made it, I won't say easy, but at least easier to prove the correctness of a program because, you already had the formal semantics of the language defined for you.

And Parnas had a team of three or four master students building a compiler for this language, compiling to some sort of pseudo code. And since I want to do a theory dissertation, and Parnas was interested in proving programs correct, what I did was look at how to prove the correctness of a compilation. That is to say, look at the semantics of the source statement and look at the semantics of the generated code or actually more precisely code templates and show that they actually matched. And this would verify not the correctness of the compiler, but the correctness of the compilation.

And this is an all theory dissertation--all proofs, no programming. Arguably, I should have implemented some of my ideas. I did find one mistake in my attempts to prove it, I did find one mistake in the generated code. So regrettably, I found that just by inspection, not by actually proving that it was incorrect, but that was what the dissertation was about.

Yost: And in doing this and focusing on formal methods, were you thinking about computer security at that time or not?

Bellovin: I wasn't. Although in retrospect that work does have computer security relevance, I just didn't realize it at the time. My first computer security incident happened around 1971. It might have been '72. I no longer remember when. I was working at City College of New York. There were a couple of jobs submitted, again, a batch system, punch card batch system that seemed to be asking for unusual resources that student jobs had no reason to ask for. And I took a look at what was submitted, and it was a couple of students doing things. And one I hired to work for me and the other I sent to the dean because of what it looked like their

programs were trying to do. So that was my first security experience coming from the system administration side.

The security relevance of my dissertation, which I said I did not realize until 12 to 15 years later, is that if this had been implemented, it could have caught Ken Thompson's famous compiler trap from reflections on trusting trust because the semantics of the generated code would no longer match the semantics of the source code. So, in principle, it could have caught that, but I didn't realize that at the time. I was not thinking at all about security. I was thinking about correctness. There weren't a lot of people thinking about security. I won't say none, but there weren't a lot.

Yost: Right. So, you discussed Brooks and his influence on you as a mentor and his support. Can you talk a bit about David Parnas and his mentorship?

Bellovin: Parnas was over committed and didn't give me as much attention as I, in retrospect, needed. And in this time and place, most of the doctoral students in this department did not publish papers on their way towards their dissertations. They worked and they wrote, and they worked, and they wrote, and they would show what they had to their advisor. And I'd go up to him and say, "Dave, I just finished Chapter 3. Do you want to look at it? You have any questions?" "No. Well, if you are happy, I'm happy." This was a mistake. He was a full professor. I had never done a dissertation before. And when I handed him the finished product, he didn't like it. He did not think my proofs were nearly rigorous enough.

So, it took me a while longer to rewrite the proofs to his satisfaction, by which point he had actually left Chapel Hill for other places. It's another thing I have learned from, but just again, in a negative sense, I make it a point to meet with all of my project students, undergraduates, senior thesis, master's projects, master's thesis, doctoral students for an hour a week plus more if they need it. I'm going to keep my hands on what it is they're doing and correct them when they're too far off.

After we're done today, I have got two calls scheduled. One with a former visiting undergraduate who's gone back to her home country but wants to continue the research she was doing with me,

and one with a master's student. So, I try to get my students all the time when they need it and make sure that I can head off mistakes, head off bad outcomes before they become traumatic.

The other person during my undergraduate career who was influential was Brian Kernighan. And he came through on a book tour. He just published a book. And since that was the year that I was a faculty member, I got a private meeting with him. And he didn't realize just how much his ideas had impressed me. Now I know him quite well from our time together at Bell Labs and such, but he didn't know until about five years ago just how important his ideas were to me, as I developed in my career.

And then of course, there was my work developing USENET, which was completely student led. And there were security components to that. But mostly we did not know what we're doing from a cryptographic perspective. We're not going to promise more security than we can deliver, so we're not going to implement it.

Yost: And before we jump to your concurrent work on USENET, is there anything else you wanted to say about your time in graduate school besides USENET?

Bellovin: I said I had a long, strange career there between my year off as a faculty member and this interaction near the end with Parnas. I was a doctoral student for about 10 and a half years for a long time. I held the record for longest time there by someone who did finish. I took two years off to work full-time in the area on what was intended to be one of the early Unix ports. That's a whole separate story before going back to work full-time and ultimately finishing my dissertation. Yes, there was personal stuff in there that it's not really worth going into. But I'll give one more anecdote about Brooks.

One of the things Brooks is best known for is something called Brooks' Law, "Adding manpower to a late project makes it later." Well, I had been a graduate student there for about two years. And the spring of my second of my second year, he came up to me and said, "Steve, what are you planning on doing this summer?" "Well, Dr. Brooks, I would like to teach. I had never taught before." And he said, "Would you like to work on professor so-and-so's

project?" "Well, I would rather teach." "Well, he's got deliverables in mid-August, and the project is running late, and I think you can help." "Dr. Brooks, are you going to try add manpower to a late project?"

[Laughter]

Bellovin: He laughed, said, "Do it. In the fall, you can have whatever you want." And he was right because this was the exception. Brooks' Law is really about the amount of communication. And what had to be done was to port this code base written in PL/I for OS 360 to make an interactive system in PL/I on a Multics system. And know as well that two versions of the same language are not necessarily going to be quite the same. I needed to make it interactive, I needed to learn Multics and so on.

Well, he knew that I was really good at learning operating systems and porting code and so on. And so, I was working independently of the rest of the team working on that project. So, that was my first experience with the ARPANET for a summer, dialing up remotely to this Multics system at the Rome Air Development Center. And we got it working on time. And yes, he wasn't a fundamentalist in Brooks' Law. Let's put it like that *[laughter]*. Wasn't a literalist. He knew what it was about *[laughter]*.

Yost: So, moving now to discuss USENET and its Unix-to-Unix dial up network architecture, that, the idea for that, came from Tom Truscott and Jim Ellis?

Bellovin: Right.

Yost: And that was at Duke in 1979. Can you talk about your earliest memories of the project and when and how you became involved?

Bellovin: Okay. So, to make that life really easy on that, because about four and a half years ago, I did a series of blog posts on the early history, which I just in the last week finished writing up as a formal paper. I'm going to proofread and submit that in another week. But yes, it's a subject I have been thinking about a lot.

Duke was ahead of us, ahead of UNC on Unix. They helped us bring up our first Unix systems around 1976 or so. And most of our computing was done on the IBM mainframe that the

university's computer center ran, or at TUCC, Triangle University Computer Center ran for universities throughout North Carolina, but Duke helped us bring it up on a small [DEC] PDP-11. They had a PDP-11, a much larger PDP-11. And the 7th Edition Unix had just come out.

Unix in those days was very much a cooperative, share your modifications to software, self-help, sorts of things, completely unsupported product from Bell Labs. but essentially everybody who got it had a source license, which made it really easy to change the operating system, share your changes and so on. And one of the popular places to do this was what was called the Unix Users Group.

Duke had installed a modification, which they got from someplace, I don't know where, to print out once a message at login to a person. After you saw that message, you would never see it again. It wasn't the standard message of the day because you're dealing with terminals that were running at maybe 15 characters a second, and you didn't really want to see a long message multiple times. It was just too slow.

So, they liked this modification, but it wasn't compatible with the login command on the 7th Edition. And they started saying, "Well, we could try to port this mod, but it's annoying to do so, and maybe we want a broader discussion system." So, Tom and Jim called a meeting at Duke, invited a bunch of people. I was the only one from Chapel Hill who attended. A few other people from Duke, and we designed that, NetNews, at that meeting. And the original design was that it was going to be a star network with Duke as the star. It was a peer-to-peer network, inherently peer-to-peer, inherently distributed, but the initial operating concept was as a star. And the reason why was that modems were comparatively unusual.

Autodial modems were extremely rare. Why? Because in that day and age, you couldn't just buy an off-the-shelf auto dial modem. You had to rent something from the phone company. Phone companies still claimed that weird devices were going to damage their network and endanger their personnel. So, you had to go rent a dialer from the phone company and a special extra device in your

computer and so on. It would be an expensive process. Very, very few places had an auto dialer.

Duke had a home-built auto dialer. I like the idea enough that I worked with my department's electronics tech to build our own, basically abusing the control signals on the serial port to do software timed pulse dialing. And it was out of spec, but the implementation specs are actually pretty loose. You could reliably do pulse dialing with software timing even with the coarse timers of the day. Both schools had home built auto dialers. So, the concept at Duke was they would dial everybody and be reimbursed for their phone calls. Again, this day and age long distance calls were quite expensive based on distance, time, et cetera. So overnight we'll call you and relay news.

So, we worked out a lot of the design, multiple news groups and so on, and we also settled on the name. And the way I put it to this draft paper, Net News is the name of the technology. USENET was the name of the particular network. And where did that come from? The previous meeting of the Unix users' group, the Bell Labs lawyers told us that the group could no longer use the name Unix in its name. That was going to infringe on their trademark. So, the leadership renamed the organization to USENIX.

And we said, okay, we were going to call our network USENET, the same kind of homage to users but also to Unix and to USENIX. And we hoped that this would become kind of the official network of USENIX, and USENIX would use it to distribute information and such. So, there's a little bit of an inside joke there. It was Jim Ellis who came up with that name. We all thought it was great. So the communication protocol was going to be UUCP, the Unix-to-Unix Copy Program because that came out the 7th Edition Unix. It was designed to work with auto dialers.

I wrote a device driver for my home-built auto dialer that presented the same API as the official auto dialer device driver would, so you didn't have to change UUCP in that regard. It was all going to just work. And I wrote a shell script. It was about 150 lines of shell, of Bourne shell, which was new with 7th Edition Unix. And it had multiple news groups. It had cross-posting; it had distributing to other sites. And because it was a shell script, it was very easy to change and try something different. When you're trying to

prototype, that's a really good characteristic. I could have written it in C, of course, but C is not a friendly language for character string-oriented programming. And our PDP-11 was so small and slow that doing a compilation was going to take a noticeable amount of time.

My shell script was pretty slow running on that machine, but it was a lot faster than doing new compilations would have been. The original prototype was about 150 lines of shell. Unfortunately, that script has been lost to time. About the early 90s, I looked around through everything I had from it. I did not find a copy of it, which is too bad. I know how it worked, but I do not have a copy of that anymore and made a number of design decisions based on what was my traffic estimate of USENET would grow to 50 to 100 computers maximum ever, and one to two articles a day, most of which would be about the same sort of Unix self-help that we have been practicing with using. So, for that matter I was familiar with, from the IBM world with SHARE the IBM users' group where sharing software was very much a thing.

So, it was engineered for that sort of load. And there were a number of design decisions that were fairly dubious in retrospect, because my traffic estimate was off by several orders of magnitude even two years out. But it was a great way to prototype. And it's not that we didn't think about the engineering. It's that our traffic load estimates were very, very wrong. And we knew there would be a need to control the network. We contemplated using digital signatures, cryptography to do this, but we had no idea how to do it.

We were all familiar with Public Key Cryptography. We had read about it in Martin Gardner's Mathematical Games column in *Scientific American* in 1977. We had seen the original Rivest, Shamir, and Adleman paper Communications in the ACM a year later. We knew all of that. What we didn't know was how to engineer a system like this. We needed certificates, which had been invented, though we didn't know it at the time, at an MIT senior thesis by Loren Kohnfelder and how would, in a pre-internet era, were you supposed to learn through the existence of a senior thesis at another school, whose existence you didn't even know of.

There were no cryptographic hash functions. There was no certificate that we knew about. We concluded that we did not know how to engineer the system, so we left that out. Now, today, I know far more about cryptography. I could do it even with the technology of the time, but back then we didn't know it, and we knew we didn't know it, so we left it out. And we thought it would be wise because the export control regulations were in effect an exporting cryptography.

I would have had unpleasant conversations with federal law enforcement. Let's put it like that *[laughter]*. Also on the other hand, this code would have been widely distributed before the patents on RSA and public cryptography were patented. And this code would have been all over the world before the patents existed. It's curious to speculate on what would have happened after that. So that was something we left out deliberately. Not that we didn't know that it was a serious omission, but we knew that we didn't know how to do it correctly.

When we had the protocol down straight, I rewrote my shell script in C and still had one major limitation that it had a lot of local news groups and only one that was distributed called Net. And that was undesirable. Steve Daniel at Duke rewrote it to have multiple distributed news groups, multiple hierarchies, and that was the version that was ultimately distributed to the world. It was announced in January 1980 at a USENIX meeting in Boulder, Colorado.

And it started slowly, grew slowly until Mary Ann Horton at Berkeley started gatewaying some ARPANET mailing lists into USENET. USENET had a chicken and egg problem. With no content, there was nothing to attract users, but without users, there was no one to generate the content. While these ARPANET mailing lists were a source of content, this attracted people who generated a lot more content that was native to USENET. And after that, growth took off very, very rapidly. Ultimately, Horton and a then high school student named Matt Glickman had to rewrite it as B News to handle many of the deficiencies from my very bad assumption about load.

Yost: What about funding and sponsorship? Were Duke and UNC contributing much? Were there others? And what about grants or sponsoring agencies?

Bellovin: When CSNET came out a year or so later, we joked that when professors come up with an idea for a network, they go get an NSF grant to fund it all. We were graduate students. We just did it. In retrospect, there was a bit of faculty support at UNC. It's not that no one knew what we were doing. I think, in retrospect, there was more faculty support from Duke. Duke was going to have to get reimbursed for these long-distance phone calls, and that had to be arranged to the department's accounting system. I now know that, though I didn't think about that at the time. So clearly there was faculty on board.

At Chapel Hill, they were willing to put up with a few phone calls and buy a better serial port for our Unix machine but there was no funding involved. One of the other important things that happened is the network did operate as a star, but not with Duke as the central node because the billing and traffic may have just made that impractical. Bell Labs Research was the central star node. Tom had interned there one summer and then a group at DEC picked up the load.

There was a group at DEC responsible for interfacing with the Unix community. Bell Labs PDP-11s and VAX are mostly run in Unix, and they were a huge customer of DEC. So, DEC did have a support group for this nominally unsupported operating system on their hardware, because they did not want to alienate one of their biggest customers. And this group persuaded their management that running up these huge phone bills or long-distance phone service was worthwhile. So, they picked up the load. There were a few others who picked up the load eventually as well. But no, there was no official faculty sponsorship at Duke or UNC beyond a very minimal thing. I said, on my end, there was a serial port, and with that, a few phone calls at Duke's for the billing system, such as it was. I don't know any of the details of that.

Yost: What were the thoughts of you and the team members about what was happening with personal computers? Of course, the Apple II was a year before, IBM PC a year after USENET got started.

Bellovin: So, I personally tracked all of this, but did not actually buy something of my own because I joined Bell Labs in late 1982, I had better access, at work, to much better computers than anything I could buy at the time.

Bellovin: The early PCs just were not sufficient for my needs. Having Unix, I did not want to go back to a lesser operating system like CP/M or DOS. The early PCs didn't have memory management or kernel mode, didn't have enough RAM. Most of the early ones didn't have discs. So, I follow this as a "where is this technology going and when does it make sense for me to buy one?" But that time didn't come for a very long time for me. Because again, I had first dumb terminals, then a very smart terminal, bit mapped terminal and then my own SUN workstation at home. Why did I need a PC that wasn't as powerful and couldn't do as much?

You could have had a later PC as a USENET node, though it would have needed a fair amount of disk space. It could have been done, I suspect, by the late 1980s. You had larger disks. If you only received a few news groups, you could have done it. But personally, there was no point to that. In fact, through the '80s, most USENET users were students, often graduate students in Computer Science departments or employees of medium to large corporations. And that actually was one of the things that constrained people's behavior. If you misbehaved, there's going to be a complaint to your management or your department chair or dean. And that was a fairly powerful incentive to behave yourself.

So, there weren't too many problems through the 1980s. I won't say none, but there weren't nearly as many because of the functionally restricted access. All this software ran on Unix, If I went to run on PC, I would have needed a UUCP port to this PC. This was the environment you were fighting against. The time-sharing systems were much better suited because they were bigger, and they were running Unix already, and built the gateway, sure. But why? If you had enough access to build and run a gateway, you didn't need it.

The PC revolution, it wasn't until Windows 95 with TCP/IP support, which is a fairly late addition as a standard thing, that PCs were really—call it first class systems on the net. AOL was client only, client only sorts of access, same as CompuServe and Prodigy

and so on. These were not first-class systems on the net. Once you had TCP/IP, then you could be. That was an add-on you could get for MS-DOS in the late '80s and third parties, but it wasn't standard until Windows 95. And that was after the Web had become a thing and Bill Gates realized he would better be in on this.

Yost: What about some of the early time-sharing companies and divisions of time-sharing companies, Tymshare's TYMNET and Control Data CYBERNET?

Bellovin: As far as I know, most of them didn't have very much interest in using it. AOL was an exception but if I recall the history correctly, that was not until the early '90s when USENET was already quite a substantial presence online, and people wanted access, so they set up a gateway. You know the captive ones, a lot of it was trying to be the information providers, and this was going to be their value add. They were not about peer-to-peer, user-to-user computing. They were about. "we're going to provide content to you and you're going to love us for it." If you switch to a competitor, you're not going to get our content.

USENET was a commodity service. If anyone could offer it, then it's not sticky. It's not going to hold you there. They were only going to do it by popular demand. I said, to my knowledge, AOL is the only one that did. I could be wrong on that. I had much less contact with any of the others.

Yost: So, in the early years with the ARPANET, there were kind of the haves and have nots with the agencies that had support and projects. Can you speak to whether there was a sense among the team of pushing to kind of democratize computing and bring in a much larger group of users?

Bellovin: We were not in the position to do so. In order to be on the ARPANET in the early '80s through the mid '80s, you basically needed a DARPA contract, which for the most part, neither UNC nor Duke had. I had this one summer of access to the ARPANET to contact the Multics system but that was dial up terminal-only access. Some of it had to do with Brooks' philosophy of running a department. He wanted the department to be run on hard money. It's all great to get grants, but when, and almost a direct quote,

"But when times get tough, I still want you to be able to pay your bills. I wouldn't be able to pay the bills." So, there was no chance for us getting on the ARPANET.

Now, there was an interesting policy, less legal question that I had little insight into about this gateway at Berkeley. And there were two rules involved. One was that the ARPANET was for research use only. So, it would not be a competitor to private data networks. And that's why access was so tightly restricted. You did not want a government network competing with these nascent private networks like Telenet and TYMNET and so on.

And second, ARPANET users must be only for research purposes of the ARPANET by authorized ARPANET users. And someone, I don't know who, was persuaded that this experiment in internetworking and interconnection was in fact a very valid research use of the ARPANET and therefore, this connection was allowed to exist and grow and include forwarding mailing lists and USENET postings or eventually bidirectionally forwarding email bidirectionally and so on.

So, yes, it would have been nice to have had access to some of the other facilities of the ARPANET, including, of course, the much higher speed links. But email was one of the major applications on the ARPANET at the time. And we had that, not in a great way, but we had it. And this made a huge, huge difference into whom you could talk to. I mean, this is pre-Web, this is pre-search engine. We did not have the ability to pull down files via FTP. Eventually, at least some of these sites set up email to FTP gateways, but I don't recall when that was done. That was never in major use from where I sat.

And by the late '80s, access started opening up. So, it became much less of an issue as the ARPANET became the internet and MILNET and such. So yes, there was resentment, but that was way above our pay grade, and we had the email connectivity, and we had the social network contact. I have often called USENET the first social network. And just yesterday I saw someone on Mastodon say that Reddit reminded her of USENET. And I said, yes. I was talking last week with one of the higher ups at Reddit, not about the current controversy. And he said that one of the

things that he liked about Reddit was it reminded him of USENET in his earlier days.

So yes, we did not foresee the social use of the network. We saw it as technical and maybe functional. So, self-help for Unix users, department administrative announcements, maybe things like used car ads within your locality. We explicitly talked about that as one of the possible use cases. But the fact that you would want to chat with random strangers about raising your kids or argue with them about politics or what have you, that was not on our radar.

Yost: Was Minitel on your radar at all?

Bellovin: I don't recall when I learned about Minitel. if it—I don't recall when I learned about Minitel. My guess is that it was somewhat later because I don't think the technology of '79, '80 could have supported Minitel in any reasonable form. Maybe it could be using a TV as your screen, certainly some of that. But I don't recall whether there would have been a Minitel and Teletex systems and so on.

Yost: Obviously this is a time before file sharing with music and films. But text could be shared. And were there discussions about copyright and material that might be shared that could be in violation of copyright law?

Bellovin: Not that I recall. I won't say it would have been impossible, but it would have been hard. Consumer CD players didn't come out until '82. It was probably 10 years later before you had CD-ROM drives on computers. So, most people had no way to get music onto their computers. Scanners of personal size didn't exist. JPEG didn't exist. MP3 didn't exist, let alone MP4. So, you want to go share a book? Got to go retype it. I won't say that can't happen.

In fact, I know of one case in the '90s where that did happen. But the only thing that could have been shared improperly and easily was Unix source code. But again, at the time, almost everyone running Unix had a source license, so you're only sharing it with other people who have Unix source licenses. So, there was no violation of your license in those terms. So, no, I don't recall any discussion of that.

It's not that I was unaware of proprietary software apart from the Unix license. It was during the '70s that IBM started switching to a much more closed source proprietary model for its software and unbundling their operating system and other components from their hardware. So, I was aware of this, but we never really talked. I don't recall that we ever talked about this. And I, again, don't think this actually became a real issue until the mid '90s.

Yost: So, there was some important terminology and associated developments with what we later understand to be significant issues with social networking, "flame," "sock puppet," and "FAQ's," comes out of this. I would like to ask you about each of these individually. So, could you start with *flame*?

Bellovin: Okay. Yes. So flaming is an old concept. I suspect the word came from the ARPANET, though, I don't know. But when I joined Bell Labs, late '82, I had a meet and greet meeting with my boss three levels up and I walked in and he said, "Hi, Steve, I have seen your flames on NetNews." And this was a revelation to me that *[laughter]* management three levels up were aware of this, was reading it and recognized my name. So, I had a very early warning that I really needed to watch what I said online because it could come back to haunt me.

So, flaming took place. Flaming as a word, was very early. I said, given how early it was, it quite possibly did come over from the ARPANET, but I don't know. Trolling was also an issue. fairly early on, someone at Bell Labs got annoyed at something or other, and created a news group called net.suicide and said, "Members of the net.suicide club were going to meet on the roof of such-and-such a building." Everyone regarded that as being in bad taste, but actually turned into a real forum for discussing suicidal tendencies and so on. It started off trolling.

The same individual later wrote one of the first bots. It would take text and use a Markov model to generate random text that matched the Markov model of the existing text, produce things that sort of almost made sense. And you weren't quite sure if this was a disturbed individual or a bot, or what. It was quite remarkable. It was being posted in the name Mark V. Shaney as from Markov Chain. And again, this was in the 1980s. You can call that a sock

puppet or not as you choose, but trolling and flaming were there early on.

The two most disturbing early interactions. At some point in the ‘80s, someone who I have no qualms about identifying as a neo-Nazi, he’s still prominent in neo-Nazi circles, started showing up and tried peddling his brand of garbage. And he was basically shamed into going away. No one wanted to hear from him.

The second, also driven off by community pressure, was someone who identified himself as an executive of an organization called NAMBLA, the North American Man/Boy Love Association, a pro-pedophilia group that became infamous in the ‘80s. And he started hanging out on the newsgroup misc.kids where we talked about child raising. You can imagine how popular this was. And again, he was forced to shut up and go away. Now, he could have continued reading it. We have no way of knowing this. We had no way of controlling this, but he at least stopped posting.

Spam didn’t become a problem until the ‘90s with the very famous [Laurence] Canter and [Martha] Siegel episode [first major commercial mass spam on USENET]. By that point, I was no longer actively in a “management” role of USENET. Through the mid to late ‘80s, a governance structure grew up called the backbone cabal. And this was mostly the systems admins of the handful of star nodes on the network and they set policy for the network. Call it an honorary member for me. I was not relaying traffic for anybody else, but as the only creator of USENET, still active on the network, I was an honorary member of, and participated in the discussions.

And the backbone cabal pushed through what was called the great renaming, reorganizing the hierarchies. It was basically its last activity because it was not governing with the consent of the governed, so it had no particular mandate, and it was forced, basically, to disband as a governing body. But while it existed, it would do things like create news groups, cancel offending messages and so on. And later on, canceling offending messages happened by individual system administrators, and cancelling bots. And you get war in cancel bots because of the lack of authentication that we worried about back in 1979. Didn’t know

what to do about it back then so those were the abuses we first saw.

You know, one of the lines that was in the original announcement from January 1980, was about abuse. You basically had what I call FAQ in the original announcement, "may there be abuses?" Yes, probably, but we don't know what they will be. Let's build this, let's get it running, then we will learn what the real problems are and how to figure out how to fix them. We can't put it better than that. We knew there would be trouble, we just didn't know what would actually require a technical or social fix. Well, we have learned.

Yost: What about gender? There are many more males in computer science and probably men were the great majority of the USENET community. But were many women participating and do you think they felt comfortable participating?

Bellovin: I don't quite know how to answer that. I don't know that I would have been sensitive enough then to those issues to have noticed that. On groups like misc.kids, there was certainly a large number of women participating. If they were shouted down or if there was a) mansplaining going on, I don't remember it; and b) I don't know that I would have noticed it back then. I'm the wrong one to ask that question.

Were there women on USENET very early on? Yes, absolutely. One of the first people I met when I joined Bell Labs in '82 was a woman who was very active on USENET. Another couple I know met that way, and they got married and had kids so that existed. When the great renaming happened one of the early news groups created, and I guess this would have been about '86, '87 was social chat, that MOTSS, Members of The Same Sex for the gay community.

So, at least it was accepted well before the broader society in this country was going to accept it. Were there sites on USENET that declined to receive that group? Probably, but it was a decentralized network. Every site could make its own decision about what it wanted to carry. I'm unaware of anyone who blocked that group for that reason. It doesn't mean it didn't happen. This means, I don't know. Again, by that point, there were probably thousands of

nodes on the net, and I would have no idea what most of them were, and what their policies were. Certainly, very many hundreds, possibly thousands.

Yost: So, in 1982, you defended your dissertation and started work at AT&T Labs. Can you talk about what led you to join as a Technical Staff member of AT&T Labs and did your work with USENET continue to a degree?

Bellovin: My work with USENET certainly continued. I had offers from IBM Research, and from Bell Labs, and Bell Labs was this storied place. It was kind of a dream environment for me. Systems programming at Bell Labs, Murray Hill, gave a lot of flexibility about what I was going to do other than keeping the machines running. When I got there, there were three lengths of Ethernet cable running TCP/IP in the whole of AT&T, which at that point was a huge, huge company. The breakup order had not yet gone into effect from Judge Green. So, it was a huge company. There were three lengths of Ethernet, my lab, another lab at Murray Hill, and a cable connecting the two labs.

So, I took over one and a half of the first three pieces of Ethernet in all of AT&T. It was running TCP/IP, which I had learned about previously through my reading while I was in Chapel Hill. So, I knew what was going on. And I continued working with it and I helped extend the network. Horton also showed up at Bell Labs. And together we helped spread TCP/IP throughout Bell Labs. When running it, one cable connecting the two labs at Murray Hill, grew into what we called the Murray Hill Backbone, interconnecting many different labs. In those days, there were no dedicated routers. If you wanted a router, you stuck a second Ethernet board into a Sun or VAX and said, “You’re a router. Forward these packets.” And it was very easy to make configuration errors that would take down a network.

And I started around ‘85, ‘86. I said to myself, if somebody could send these packets out to take down the network by accident, what if they do this deliberately? And that was the start of my transition to becoming a serious security person. I had had a few other things. I mentioned this incident when I was in college. When I was in graduate school, I had read the Morris and Thompson paper on password guessing, and I implemented a password guessing

program, found a friend's password. As a system administrator, you have to be thinking about security.

One of the things I did was, since I was running a lab with what at the time it was called a super minicomputer. I said, "let's take a small one of these and make this our gateway to the world." There were a lot of reasons for it, like having one email host for the whole lab. But security was another piece of it. The other computers in the lab didn't have to talk to the outside world. So, I could lock down one machine and not worry nearly as much about the others.

But I said the real transition happened in a few other minor system administration type things that I did that were security relevant. But the real transition happened when, yet again, that went down because someone had misconfigured a routing protocol. And that's when I said, what happens if someone does this intentionally? And that's when I started thinking seriously about security. That leads me to probably guess this is my second published paper. It's still a very important one, "Security Problems in the TCP/IP Protocol Suite", which shortly before that paper came out, there was the internet [Morris] worm of '88 which we were almost hit by, but not quite.

We didn't have our formal link to the internet, yet that was coming. That was going to be through a security gateway that I was partly working on, though, not mostly. But we had a link to another company, which was on the internet. And it could have spread to us, except for one accident. The internet worm could affect Sun 3s and VAXes running I guess it was 4.3 BSD, Berkeley Software Distribution. And the people at the other end of this link were running Ultrix, DEC's product version of Unix, which was very similar, but not identical to 4.3 BSD. It was not quite binary compatible. So, the binary that was injected for the worm crashed there, and hence, never had a chance to try to attack our machines. So, it was fair amount of luck that went into that.

And one of the reasons we had this link is I wanted to start experimenting with what today we would call firewalls. So, I was already trying to move full-time into security by '88, and I got my management to pay for this link and so on. At that point, network security was my biggest concern. I then did an internal internship

at Bell Labs at what was called the Unix System Development Group. The product arm I was shipping, I was developing product versions of Unix and selling it. And that was my first real experience in a real-world software development organization.

I have learned a lot of things about how things were done in the real world. Yes, I had had the academic explanations from Fred Brooks in ‘73. And here’s how it was being done in ‘88 or ‘89 in an operating system development organization. And one of the things I worried most about was security. And there was an interesting incident that occurred during my time there. I opened my morning newspaper, opened *The Times*, and saw on the front page that three people had just been arrested and charged with hacking. And one of them I knew because he was an employee of a company that had joint development agreement with AT&T. It was Sun Microsystems. That’s pretty obvious to track down.

He was an employee of Sun Microsystems. He was working on this joint product, and he had just been arrested for hacking. This did not give me a warm, fuzzy feeling. So, I wrote a memo to management saying that we have to go audit the codebase here because we don’t know what he did. So, management gave what I now recognize as the appropriate typical response. "Great idea, Steve. Organize a team and do it." *[Laughter]* so I got to go lead a team that was going to audit codebase, and it was an interesting exercise.

We decided based on the threat models of the time, that was not likely to be anything in the kernel. It was just too hard to do nasty things in the kernel that would persist and so on. Today, of course, that’s a real threat. It occurred in 1990 and was not a serious threat. There’s another story I’ll have to come back to in a moment. We also knew what modules this person had access to, because we were using a version control system so we could look at the check-ins to see what code he had checked in. If he hadn’t touched it, it wasn’t a threat in that sense. And we did have a structured process for committing code to the formal codebase. So, this made the project manageable in scale. He wasn’t looking at the compiler, therefore we didn’t have to look at the compiler and so on.

And we found, or I found two interesting bugs. One was a very serious security hole, but it turned out to have been accidentally

introduced by another member of the audit team. I went to her and said, “Did you do this?” “Oh my God, I did.” So, we fixed that one.

The other was more curious. It was a combination of two independent bugs. The comments didn’t agree with the code for one of them. And together, plus a common misconfiguration, added up to a serious security halt. And I’ll call this my peak “pride in debugging” incident. And to this day, I do not know if it was deliberate or not. There’s a benign explanation for why the comments might not have agreed with the code. So, on even days, I think it was benign and odd days, I think it was malicious. Today’s an even day so it was benign. It is just an ordinary bug in the code.

So yes, he ended up pleading guilty. Of the three who were arrested, one was acquitted, one was convicted. I think the third, this one I think plead no contest if I recall correctly. So that was an interesting episode I saw. But at this point, I really considered myself full-time a security person, but I just wanted this other experience.

The other interesting incident has to do with the Morris Worm, the internet worm. A few days before the worm hit, word was passed around very quietly of a serious security flaw in the FTP Daemon. And this flaw was straight root access on the machine if it was running FTPD. But it could only do that by modifying the system. Unlike the flaws that were actually exploited, this one, you could add a line to the password file as root, for example. So, it was a devastating hole that had to be fixed. But if it was used for the worm, we could only work by modifying the system.

And word of this flow was passed around very quietly in the security community, the high Unix communities. I knew who actually had found this bug and it was none other than Robert Morris. He had found it, had arranged for it to be reported and fixed because he could not use it for the worm without causing damage to the system. So, this was very strong evidence that he did not intend to cause damage.

Well, I knew Morris’ father from before he had gone to the NSA. I knew all the luminaries in research who knew him. I knew Ken

Thompson and Dennis Ritchie and so on. And I told them what I knew. I told them to report this to the defense team, to his defense lawyer. And there was a little bit of infelicity in the wording of the Computer Fraud and Abuse Statute. Morris was the first person ever charged under it. It was a little bit ambiguous whether he had to intend to cause harm. When "knowingly causing harm" as opposed to "knowingly intruding". And if the judge had ruled that you had to intend to cause harm, this arranging to report would have been extremely important evidence that he did not intend to cause harm, because it only could have been used by causing harm.

The judge ruled against Morris and the Second Circuit upheld this ruling, so it never came up in court. But I had this hard technical evidence that he did not intend to cause harm and made sure the defense team was aware of this. Was it a stupid thing to do? Sure. He was a young graduate student. He is now a very distinguished full professor at MIT, a member of the National Academy of Engineering. And he never went back to security because he didn't dare, and the security community lost a great mind as a result of this. But he did not intend to cause harm. It was very clear to me.

Yost: As I understand you encouraged Spaf, Gene Spafford to write on the Morris Worm. And he did so to great impact.

Bellovin: Yes. Spaf and I actually met in probably '83 because I was living in New Jersey by that point '83, '84, just after he had completed graduated school, before he started his job as a faculty member. And it was sort of the few USENET luminaries I had actually met in person at that point.

Yost: Go ahead.

Bellovin: You can't be a good system administrator, network administrator without thinking about security. And that became increasingly clear through the '80s. I got into security through the system and network administration, so that path before I lateraled into research. I was doing research, but I wasn't formally a researcher my first several years of the Bell Labs.

Yost: So, in the security field more broadly, a lot of the early money was from the Department of Defense. There was Roger Shell's work in the '70s and the National Computer Security Center at NSA in the

first half of the 1980s and beyond. Were you also thinking about access control systems and trusted systems and intrusion detection systems?

Bellovin: I was to some extent by the late 1980s. Listen, by the late '80s, I was doing security network and security work full-time. I already mentioned my intention to build a firewall. That got preempted when Jeff Mogul built the exact sort of thing that I was contemplating. Well, he did that. Let me move on to something else.

When we had our initial connectivity at the labs to the internet, Bill Cheswick was the one who was actually running the gateway machine. He took that over from me. That was his actual job. He was a system administrator in the computer science research lab. The same lab with Ritchie and Thompson and so on. I was in a different lab in research, and he was running this.

And I actually wrote an early intrusion detection system. I connected another interface, a second Ethernet interface on my Sun workstation to the external backbone to see what was coming at us. And this is the days of the old coaxial cable Ethernet with the transceivers, big physical transceivers. I realized that there was no way that I regarded as high assurance to keep my machine from transmitting on this link. I had read Cliff Stoll's book and so on. So, what I did is I just clipped the transmit wires. And regarding the technology at the time, this did not impact its ability to receive, but it physically could not transmit.

So, I was monitoring the traffic and I actually got two papers out of that. One was "There Be Dragons," about attacks that I was seeing that even around '91 or so. And second, I called "Packets Found on an Internet." Just weird stuff that I was seeing that really shouldn't have been there. Even in '91, a fairly small internet, almost certainly less than 10,000 machines there. Yet there was traffic that was reaching us that had no right to exist. So, I wrote this up as a paper as well. So, I got two different papers out of this effort.

Meanwhile, Bill had built what today we call an application-level gateway to the internet that would handle basically mail and FTP and anything else that you wanted to write a custom proxy for. But this was pre-Web, so we didn't have to worry about that. So, this

was the early pre-web days I was getting involved in this. I was thinking about access control and intrusion detection. I was aware of the Rainbow Series of security publications and the *Trusted Computer System Evaluation Criteria (TCSEC),* commonly known as the *Orange Book* from the color of its cover.

When the *Orange Book* had a 2nd Edition in 1985, when the internet worm hit in ’88, this was still state-of-the-art description of how to secure a computer system. Once we understood what the worm did and how it worked, I went back and re-read the *Orange Book* and came to the very mind shattering realization that the *Orange Book* features would not have stopped it because the *Orange Book* was based in the notion that you got a trusted kernel and untrusted user land. What happened at user land doesn’t matter. And the worm operated mostly without root privileges so the *Orange Book*’s concepts wouldn’t have stopped it.

So, I went to someone at the labs, who was older and wiser in the ways of security, who said, “Oh no, Steve,”—I forget it was a B2 or B3 system—"would have stopped it.” “Well, how so?” “It requires a thorough search for bugs.” Now, okay, I was not that senior at that point, but I knew even then that that was not going to fly. It just was not a concept that made sense to me. But some of that was my education from Brooks and Parnas about the ubiquity of bugs and how impossible it was to try to find all of these things.

It took a fair number of years before I let go of the *Orange Book* [*TCSEC*] style, I think. I understand now why it couldn’t have...[pause] It was designed to instantiate a particular security model, which the worm went around that security model. It was doing things that the security model wasn’t intended to stop. So of course, the *Orange Book* systems were not going to stop it. As I said, by that point I was getting very, very involved in security.

In ‘91, I came up with what I still regard as my most original idea. I was working with a colleague at the labs on cryptographic authentication. We had published a paper on the limitations of Kerberos. And I said, “well, look, if you’re monitoring the network, you’re going to pick up the Kerberos traffic, and the initial exchange with the Kerberos server is protected by a password, basically by the user’s password. And if I can launch a password guessing attack on a password file, I can do the same on

a Kerberos packet. And how do we prevent this?" And I kept worrying about this for several weeks.

My colleague Mike Merritt told me later that he was convinced that I was wasting my time, and I went to an internal talk, and the speaker was extraordinarily boring. Did not know how to give a talk, reading a slide, this is the pre-laptop era. So, my mind just wandered, and suddenly, while listening to this talk, I had the solution. So, I wrote it down in my notebook. It became the EKE protocol. And I said, it came to me in a flash of inspiration after working on this for several weeks. And Mike and I worked on for a while, published a paper a year later, after we went through the whole patent dance and so on.

And this gave rise to the PAKE Password Authenticated Key Exchange subfield of cryptography. This paper, this concept that Mike thought was impossible, was one that I came up with as the answer during this very boring talk. And this is a story I always tell my students on the first day of class. If you're bored, stay in the class, maybe you'll come up with something like this too.

At that point, I was moving into security full-time, and on the train to USENIX in Baltimore in '93, Bill Cheswick and I happened to end up in the same car of the train. We didn't know each other that well then. We started talking about doing a book on security. And we were originally thinking of a collection of papers book, and we put together a table of contents, and I don't know what, if anything, would have happened from there, except that not all that long thereafter, an editor, John Wait from Addison Wesley, paid his annual visit to me.

And the annual visits were to chat for a while, and to always ask "Well, Steve, do you want to write a book?" And I had say, "Sorry, John, I don't have anything to say." He paid his annual visit. We chatted, "Want to write a book, Steve?" "Well, John," I showed him this table of contents. He looked at it, nodded and he said, "Collections of papers don't sell very well. I bet you could do a real book. Why don't you try?" So, Bill and I said, "Oh," went off, came up with a real table of contents. John pushed back here and there. We wrote *Firewalls and Internet Security* book.

Yost: And did Rubin join you?

Bellovin: That was for the second edition.

Yost: The second edition, okay.

Bellovin: That was the second edition. The first edition was Bill and I, and came out at 1994 just as the commercial internet was exploding. One of the very last things we added to the book were about two paragraphs about this newfangled thing called the Web. There's an appendix saying, "Here are places to get software." I rejected a reviewer's suggestion that we use URLs to indicate how to get these resources on the grounds that no one would understand what they were. It was that early in the Web's existence.

But I think it was well written. I think it was a good book, and the timing was perfect, and there were no other books on internet security by anyone for at least six months thereafter. So, it sold extraordinarily well for a technical book. More important than any financial benefits, it was the publicity effect. It really made my name known in this broader security community. And that's when I started getting involved with things like NCSC [National Computer Security Conference] and giving talks all over the place and so on. So, from that time, that book came out in May 94, my life turned utterly insane.

Yost: And it also probably influenced the early development of the industry?

Bellovin: Oh, absolutely. One of the curious things is Bill and I invented what today you would call a stateful packet filter. We were not thinking of appliances. We were thinking in terms of modifying an off-the-shelf operating system. We concluded that we could not do this with high assurance, and assurance is something I did and do care about a great deal. So, we gave a talk on the notion internal to Bell Labs. There's a memo establishing this in the Bell Labs archives, which antedate the eventual patents on the stuff issued at Checkpoint.

But they set out to build a firewall that implemented this. And that concept is what took over the market, even though it was to some extent a lower assurance implementation design, though I later repented of that. It didn't have the security guarantees that we

thought that our application firewalls have, but it was far more flexible with the internet growing so fast, that turned out to be a far more important consideration. But yes, Bill and I invented this first, but never published this externally.

Yost: Great to know this. I know Trusted Information Systems, Steve Walker's Company, did some early work with firewalls. Were you in communication with Steve and TIS people?

Bellovin: Yes, there were about four or five of us who did a lot of the early work, just talks on firewalls. Me, Bill, Marcus Ranum, Fred Avolio. Those last two were from Trusted Information Systems. Brent Chapman, Brent really didn't like our book. He wanted more of a how-to, hands-on book.

And Bill and I had never intended and didn't want to write such a thing, thinking that technology was going to change. We were more interested in talking about the concepts. So, he said, "Well, I'm going to write my own book." And we said, "Great, friend, the world needs both books. That's not the book we wanted to write, but by all means, write your own, we're happy we'll be very happy to see it." And he came out six, eight months later, and it was a fine book. It just wasn't the book that we wanted to write.

Just before the firewalls book came out, I started getting involved in the IETF [Internet Engineering Task Force], working on some network issues, especially IPv6. And of course, security became part of the leadership. I became a member of the Internet Architecture Board for a few years, and then security area co-director, which lasted until I left for academe, at which point I had to resign.

Yost: Can you elaborate on your work with these different groups and committees within the IETF that you worked on and the contributions you made and then, and what you saw as the most central issues during your time on it?

Bellovin: The group that I was most interested in before I became part of the leadership was the group that produced what's now IPsec. And there were a lot of very needed debates about two different technical paths. In my opinion, the correct one won, but it was so exhausting that the key management piece, which we put off until

later, had key exchange, and I think that went down a fundamentally wrong direction in terms of complexity and layering. And some years later I co-authored a paper with six others on how we thought key exchange should be done.

But that was my main hands-on technical area, worrying about architectural security issues, especially for IPv6, which we thought would take over a lot faster than it has. By around 2000, I was named Security Area Co-Director, and my big effort at that time, and I think I succeeded, was to ensure adequate security review of every protocol in every RFC. Most RFCs through that time would say under Security Considerations, "Security is not discussed in this memo."

And there had been a mandate that there should be a security considerations section, and it was basically null. And I wanted to do something about that. So, I revitalized what was called the security area directorate, and got a bit of support from my management. I treated all the members of the directorate to lunch for several successive meetings with support with funding from my management. That's a good way to get people to show up and do work. You don't do the work, you don't get to get this free lunch.

Bellovin: And this established the process where someone in the security area directorate was going to look at every single RFC and work with the authors to make sure it was really analyzed correctly for security. Was it perfect? No, of course not. But also encouraging people to get involved in other working groups early on to try to get security designed in properly from the beginning was important. And I said that I think my biggest achievement in the IETF was getting people to pay real attention to security from the very beginning or certainly at the RFC stage. And I blocked more than my share of RFCs for inadequate consideration of security in the early days until people started doing it right.

I remember one memorable incident where I had blocked a particular protocol, and I basically got surrounded at dinner one day by everyone who had worked on this, demanding that I show them exactly what was wrong. So, I just pulled out my laptop at dinner and showed them exactly what was wrong, exactly how a particular attack could be launched. And they muttered, and they mumbled. But I was right. They had to go back and fix this. It was

an unauthenticated message that you could be used to intercept phone calls, or Voice Over IP calls under certain circumstances.

No, that wasn't acceptable. So yes, it went out. But I said my biggest contribution there was getting the real security attention paid, and I did this by leveraging not just the two security area directors, but the whole security directorate, which was people who were knowledgeable and interested in security and willing to get involved in this stuff.

Yost: Aside from pushback from authors of RFCs on particular papers, was there a broader kind of pushback from people that just didn't care about designing with security from the start?

Bellovin: Of course, strong security, that's going to hurt performance. So, there was once a suggestion, I think it was had do with an email protocol, an IMAP variant, do a very quick status check, has no email arrived, but I really don't remember the details, but this was intended to be really, really quick and it wasn't encrypted. And I showed that this leaked information that was not supposed to leak, and you had to incur the extra round trips to set up a TLS session.

They didn't like it, but under the policies that we had pushed through they had to accept it. It was just a very clear-cut answer. There's also the fight about cryptography policy, which led to this so-called Danvers doctrine, were we going to build backdoors into our protocols. And as the U.S. government wanted them in the mid-nineties, and the IETF concluded as a whole that yes, it might hurt market share, but we were going to do things right technically. And yes, there were pushbacks. There was from U.S. vendors, you needed an export license to export cryptography at the time.

And if you put in cryptography, that's going to hurt your market share. And the IETF said, "We are an international organization, we are going to do this right, technically." And it was unpopular with some people. But that was the conclusion of the IETF, as a whole at a meeting in Danvers, Massachusetts mid-nineties, I forget the exact year. The other notable change in my career happened around '93, three separate public policy issues came up that affected AT&T.

And there were only two people in research, myself and Matt Blaze, who knew the technology, and wanted to get involved with lawyers. And this was the DMCA, the Digital Millennium Copyright Act, the Clipper Chip—the backdoor in crypto systems—and the bill that became CALEA, the Communications Assistance to Law Enforcement Act. And AT&T cared about all three. And Matt and I are the only two people in research who wanted to get involved with the lawyers. But that was when I started my professional shift towards law and policy.

When I got to do this as part of my day job, the hard part was persuading AT&T management to want what we wanted. We were often successful at that, but we had to be very careful about how we did this sort of thing. And Matt's exploit with the Clipper chip was legendary. It got a front-page story in *The Times* for his finding a flaw in it [the Clipper Chip]. but that was what led to putting in a legal chapter in the position of firewalls.

I was having so much fun with this law stuff that I started looking to some of the legal issues surrounding computer security. So, I had a couple of lawyers teach me the basics of what things were going on. Spent many hours at law libraries, looking things up foreshadowing my current career. I have no formal legal training.

Yost: Can you talk about the transition from AT&T to Columbia in 2005?

Bellovin: Okay. I was starting to get a little bit unhappy at AT&T, I just wanted to do something different. I did another internal internship in the company in the Backbone Architecture Group. That one I'll call less successful because I had too many outside commitments at that point. But I still learned a lot about how the real network was built. I had worked on some of the early voiceover IP product technology in the late 90s. I was just starting to get unhappy there. I taught for a semester as an adjunct at Penn to refresh my teaching abilities and credentials.

And AT&T was falling apart. The phrase "death spiral" was being used around the company. If SBC had not bought AT&T and then changed its name to AT&T, I don't know what would have happened. But the company was not doing well financially, and

there were cutbacks affecting research. The head of research came in and had a certificate in computing from Rutgers or something. Contrast that when I joined Bell Labs in ‘82, the head of research was Arno Penzias, a Nobel Prize winner and replaced by an MBA type years later, of course.

And it became moderately hard to publish, hard to get funds, and almost impossible to talk to the press. And I went to Bell Labs in ‘82 rather than academia even though I love teaching because I do dislike anything and everything to do with getting grants. I went someplace where I could get the resources to do interesting work.

And suddenly that had changed. It was now hard to get the resources at the labs and getting harder to publish. So, I said, why am I still here? I’m in a company that’s not in a good place financially, there were certainly more cutbacks on the horizon, even if the company survived. I can’t get the resources to do work or to travel. It’s hard to publish. Why am I still here?

So, I left, a bit later than many other people. The rest of the security people had left about a year earlier, including Matt Blaze, and we all left for academia. And I guess what underscored that was an exchange the AT&T Fellows had with the head of research. We asked for a meeting. He walked in, started the meeting by announcing “If people were going to go elsewhere for twice the salary, there’s nothing I can do.”

And we explained to him that people were taking pay cuts to leave. His response was, “Oh.” If you had expected an MBA type to be good at one thing, it’s managing and understanding what’s going on in your organization. He was clueless. No, I’m out of here.

So, I went back to what matters, and was looking to join a Computer Science department by that point. And I don’t regret the decision even slightly. The SBC takeover, when it came, brought a lot of money. The company did turn around. I have friends who are still there.

But I don’t regret even slightly having left, even if I have to deal with grants, because I do have so much more freedom in what I work on these days. And access to the Columbia Library Network, which is not something I would have had easy access to elsewhere.

So that was why I left. It was the way I joked about it. I got tired of doing anything I wanted to, so I decided to do something different for a change.

But teaching, working with students, academic freedom is not everything they say it is. Because you still have to worry about grants for a lot of the things you do, as you know full well. But I can do weird things, like write the law review papers and papers on the history of cryptography and so on, which I would have had a lot more trouble doing were I still at AT&T Labs. So, it was definitely the right thing to have done.

Yost: Your first decade in returning to Columbia you were a member of the Homeland Security Science and Technology Advisory Committee.

Bellovin: Right. Yes.

Yost: Can you talk about that Committee's work?

Bellovin: It was only in retrospect that I actually understood what it was all about. The Department of Homeland Security was organized administratively into several different sections, each headed by an undersecretary. The statute that established DHS required an advisory committee to advise the Undersecretary for Science and Technology.

So, we were not a broad advisory committee advising the secretary. We were advising the undersecretary, and the import of what we actually did depended on who the undersecretary was at the time, some were interested in what we had to say. Some put up with us because they had to, because the statute required that we exist. Some were in between. And we issued a bunch of reports.

I think that most of them are labeled for official use only, so I won't talk about them. I think there are one or two things there that I did that was useful. I don't know that the committee had tremendous impact. I learned a lot. But I don't know that the committee actually helped the department that much because again, it depended on who the undersecretary was. There are a few things very positive that we did establish, that we did accomplish, I

would say. But commensurate with the effort that was put in, that's much harder to say.

The other interesting thing is, when Obama came in, took office. The White House sent out a memo saying Homeland Security should set up a Data Privacy and Integrity Advisory Committee. And here's a list of people who should be on it. I served on that committee as well as a subject matter expert. And that was, I think, more interesting and had somewhat more impact because again, this one was created by The White House to gather information. And I think it did have positive impact there.

Fourteen years later, it's a little hard to remember exactly what it was. But that was a very interesting experience. Eventually, my time on both those committees eventually expired. I have not gone back to that. I did go to the FTC in 2012. Ed Felten was the first Chief Technologist. He had created the post; he had talked the-then chair Jon Liebowitz into creating this post. And as his time there was coming to an end, he sent me an email saying, "Do you have any suggestions for my successor, Steve? Feel free to self-nominate."

It took me about 30 milliseconds to decide that I was very interested in this. And not a whole lot longer before I asked my wife, "you want to go live in Washington for a year or two?" She had lived in Washington a couple of times before. She likes the city. Our kids were out of the house.

So, we said, "Sure." So, I had a meeting with the chair. And one of the things that I had long wanted to accomplish and why I wanted to do it was that I have been interested in public policy since I was a teenager. I used to go out there and leaflet for candidates when I was 14 years old. So, this is not a new interest of mine. I said I had a legal chapter in the firewalls book in '93. So, I was very interested in doing it.

So, I went there for a year. There was a change in chair. Liebowitz stepped down while I was there, replaced by Edith Ramirez. It took a while for her to understand why there was a Chief Technologist on her staff. I made a mistake. Jon originally wanted me to stay for two years. I had been on sabbatical the previous year.

I could have gotten another two years off from Columbia on what the university calls public service leave. But I didn't really want to be away from campus and my graduate students for three straight years. So, I stayed away. I only went there for a year, and that was a mistake.

It took several months to understand what was going on, what I could and could not do, and towards the last few months I'm wrapping up and trying to hand off whatever to my successor. So, I didn't have as much productive time there as I would have liked. Two years, 18 to 24 months, would have been a much better time. I just didn't understand that at the time. I'm not really big on understanding bureaucracies, what they can and cannot do. I'm a techie, I'm not a bureaucrat.

You know what? Broadly speaking, my role was to advise first the Chair, second, the other commissioners, and third, what the FTC called the staff on technology issues. So, in terms of advising the chair, there was—and you'll forgive me for not being specific, I'm not allowed to discuss some of these things, there was a regulation, which seemed to be going in the wrong direction. It was wrong, technically.

The Chair had been persuaded that this was the right way to do things from a market perspective, but it was very wrong, technically. I wrote a memo explaining why this was wrong, and it eventually came out my way. Was this the only contribution that coming out my way? I have no idea. But it came out right after my memo, even though the Chair hadn't wanted that.

I worked especially closely with one of the other commissioners, Julie Brill, because she was very interested in databases and privacy and such. A lot of the FTC's work is on antitrust. I did do some antitrust-related stuff, but I know as much as the average educated, intelligent, curious layperson, I have no particular expertise in antitrust, but I was in the tech industry, you know.

So, for example, what is the role of intellectual property and standards organizations? When I was in the IETF, I had co-chaired the Intellectual Property Rights Working Group. So, I knew up close and personal what some of these issues were and how different standards organizations dealt with this. So, I can write

and speak intelligently on the antitrust implications of patents particular and standards.

And I also worked with the staff. There was one case they came to me with, they were investigating a company where they suspected fraud deceiving consumers. They had subpoenaed records from this company, which turned over basically 4 million lines of spreadsheet output log files converted to spreadsheets. And this is a bunch of lawyers who had no idea what to do with 4 million lines of log file records.

And, okay, give me the following kind of technical access. And I wrote some code to crunch the data to reduce it and produce some graphs. And when you looked at the graphs, that was fraudulent and the nature of the fraud just stuck out like a sore thumb. It was very obvious what was going, it wasn't the fraud they suspected, but it was very clearly fraudulent. And they said, look, "you can't take this to court without a statistician looking at it." But you could see this from this graph as well as I can, just what they're doing. And the eventual outcome was very satisfactory from my perspective. So those were the sorts of things that I did there.

Now, I also got education. I also learned things. There was one internet thing that I regarded as a serious antitrust issue, and I had sat down with Jon and the head of the Bureau of Competition, and I explained what the issue was, and they asked me questions I had never thought of asking, and eventually I concluded that "no, there was not a real antitrust issue." Okay. I was actually glad for technical reasons, but they concluded that there was not an antitrust issue. They do antitrust law. I did not. I knew the technology. So, this is not an issue that could have been settled from just the law side or just the technical side. It took both. So that was the kind of thing that I did.

Yost: In 2016, you published a very important book *Thinking Security: Stopping Next Year's Hackers*. And it takes a very holistic approach in looking at the human element and looking at policy, and implementation very carefully as well as the technologies employed. Can you talk about the context of that book?

Bellovin: So, one of the lecture courses I most enjoyed teaching at Columbia, it was called Security Architecture and Engineering. It

was a course I created, and it was fundamentally about an issue that had concerned me since the early ‘90s, which is how do you think about security? How do you think about putting these different things together? In particular, technology’s changing all the time.

Think of when I wrote that book, the iPhone was less than 10 years old. And it completely changed the security posture of many companies. What is coming down the pike next? And then there was the firewalls book that happened through these two coincidences of Bill and I being on the same train, and then my editor stopping by shortly thereafter, this one happened through another coincidence.

I had a sabbatical coming up before I went to the FTC. I said the year before I was at the FTC, I was on sabbatical, and I had originally planned on going abroad, but I had medical adventures instead. So, I wasn’t going any place. And I got tired of seeing a lot of misconceptions about authentication. And so, I decided to go right up how to think about authentication.

And then I said, "Well, gee, there are a lot of other things that people are not thinking about correctly either." So that’s where that book came from. The way I phrase it is, “How do you think about technology change in the security business when technology changes, when your enterprise changes what it’s doing, how do you think through the issue?”

So that, of course, has been a lot of what I have been teaching to my students is, how do you think about security rather than cookbook sorts of things? how do you think about it? And that’s what that book was intended to be about. I am just barely starting on. I’ll call it a companion book, which I’m calling *Designing Security*, which is, again, going back to something that’s consumed me to this point for 30 years.

If you have a large-scale system, how you separate the functions can have a big impact on the overall security of your organization? Buffer overflows, SQL injection, that’s low-level program. We have got to go fix that, of course. But we also know that we’re never going to get all of those things right. They’re always going to be security flaws in some system. I limit the damage.

So, I'll give one simple example. I regard webservers as very fragile creatures because the production webservers are running these large complex scripts to implement all the interesting functionality. It's not just handing out a static file the way they did once upon a time. Well, these are programs, these are outside-facing programs, not written by security people. Why do you think they're secure? Well, I don't think they're secure. And I therefore regarded web servers as very, very dangerous things.

So how do you do authentication? If someone is logging into the web server, the web server does the authentication by itself, that means it has the authentication database. So, when the webserver falls, not if, *when*, the whole authentication database has fallen. You can just download it and run password cracking or whatever all you want, some sites don't, these days less so, but not that many years ago wasn't uncommon besides to keep passwords so they can send you password, reminders, password recovery. A thoroughly bad idea, but that's a separate issue.

So, I said, okay, we have this password database, which is very sensitive. It's got to be consulted by a very fragile component. This tells me that I want to go. I move the password database to a separate service, not an SQL server sitting on another computer that's being queried, because if it's speaking SQL, the whole thing's available anyway when the web server falls.

But a simple-minded service, and I'll skip two-factor authentication for now. A simple-minded service that says, "Here is a username, here is a password. Is this valid?" Or "Here is a username, here is an old password, here is a new password, please change it." And those are more or less the only functions that this server has to do. It's small and simple and stupid, and the messages are sufficiently simple that I think that they can be parsed reliably, correctly, securely with ordinary care.

Yes, just watch out for buffer for length checking. It's about the only way you're going to get that wrong. And this separation of function means that when the web server falls, your authentication database does not fall. And this is a huge step forward for your overall system security. So, by splitting up the authentication

function from the web server function, you have increased the overall security of your organization.

And I have labeled the language that the two speak to each other as Newspeak from *1984*: the language where you couldn't think a disloyal thought. It's a simple enough language that it's very hard to get this wrong. I'll give you another example here. Take a user profile database that's got your credit card for an e-commerce site, that's got your credit card number in it. We have seen many massive compromises of credit card databases. Supposed though, that the credit card numbers, and most other very sensitive pieces of the profile are encrypted with the user's password or the hash password or what have you.

This means that the attacker who compromises the web server and dumps the user profile database doesn't get credit card numbers for free. They get encrypted ones, would password guessing work on that? Sure. But the speed of compromise of the credit card number database is now limited to the speed of password guessing. It's not all there for free in one operation. It's not order-one.

And this, by the way, also gives users more incentive to pick good passwords because their own credit card data is being protected that way. It's not, "Here are the hoops I have to jump through because the website insists on it." It's my data. And yes, you're probably not liable financially, but dealing with the stolen credit card number is a nuisance. You know I had to deal with that as recently as yesterday. So far, it's only cost me about a half an hour of my time. But while we have been talking, I have gotten two email messages from my provider saying, "Okay, we're investigating this further."

So, this notion of separation of function is, I said, I have been thinking about this for 30 years. I now think I know how to write the book. And it's a long story. But curiously, I had the insight on how to write the book for looking at a law paper, which was not the sort of thing I expected to happen.

And in recent years, to anticipate what you might be asking, next, much of my attention has shifted to law and policy. So, I spent at least as much time now writing on legal topics as I do on technical topics.

Yost: You certainly did anticipate my next question or two. Can you talk about what has intrigued you most about network security and law?

Bellovin: I wouldn't so much call it network security as technology in general.

Yost: Technology and law.

Bellovin: I have had a longstanding interest in privacy. The very first course I taught at Columbia was a seminar course in anonymity and privacy. So, it's long been an interest. I started doing law for the same reason I started doing anything else, because it's fun, and it's just become more and more fun. So, my general approach to these questions is, is there something that I know that lawyers, judges, legislators don't know but need to? And sometimes it's a direct security thing, but that hasn't actually been the main focus of my legal writing.

So, my first serious law paper is actually moot because of the Supreme Court decision. As an academic, I'm disappointed, as a privacy advocate, I'm happy because I think the Supreme Court went further than we did. It has to do with the privacy of location data. And under a Supreme Court ruling from 1983, no warrant was needed to track your location in public.

This struck me as wrong because location data is very sensitive. You can learn things from it. And what we proposed was when a machine learning model can accurately predict things or accurately enough predict things that were not directly observed, then you have a privacy violation. And this is what's called the Mosaic Theory of the Fourth Amendment. No one data piece is a privacy violation. With all of them together, like the tiles in a mosaic, you have a very full picture, and it's reasonably popular among academics, it's decidedly unpopular among prosecutors.

So, when you can start predicting these things without observing them, then you have a privacy violation. And so there are four of us who worked on this. There was a law professor who is now dean of the University of Maryland Law School, a junior faculty member at the time, but we started on this about a dozen years ago.

A then-PhD student of mine, who's now a CS professor, who's also a lawyer in Germany, but a member of the California Bar. And he was a then colleague of mine, a machine learning specialist who had worked with location data. I was the least qualified person except for the fact that it was my idea, and I was the only one who could talk to everybody else. And it just went on from there. I have had two of my law papers have been cited by appellate courts. One in California, one in New Jersey.

The one in New Jersey, I'm very proud of myself, my usual co-conspirators on these, Susan Landau, Matt Blaze, and a law professor from Texas. And that's on defendant's constitutional right to access a source code that's used to produce evidence against them. A couple of years ago, the intermediate appellate court in New Jersey said, "Yes, you have this right." It's just like the right to confront the witnesses against you, and software can be buggy, and this is the only way to challenge it properly. So, this is fact. Another appellate court in New Jersey just upheld this just within the last week or two. So, this is binding law in New Jersey. I'm very, very happy about that.

So, I just keep working on these law papers now. I have three in progress right now and at least another few planned. So, I'll just keep doing this. And it's not security per se, it's many of them, but not all are privacy related.

There's one that I co-authored that has been passed around in Washington. It's me, another longtime computer security person named Adam Shostack, a law professor at University of Colorado, and two of his then-students on voluntary reporting of near-miss computer security incidents modelled on the Aviation Safety Reporting System, and what are the legal implications of this.

Computer networks are not airplanes. This is not security, this is not privacy, but it's related to security, and it's mostly a question, things like liability and disclosure and so on. And I am told that this has passed around in Washington when the Biden stood up his Cybersecurity Safety Review Board. It's an interesting step forward. It didn't go as far as we wanted, but certainly a very big step forward. So we'll have to see where that goes.

So, I write on a variety of different topics. The paper I'm most fond of was on internet metadata under US law. The contents of a phone call are strongly protected. The number you dial, the number they dialed you, are weakly protected. It was based on a pair of Supreme Court decisions a dozen years apart. How does that work on the internet? What is the equivalent of the numbers that you dial?

Obviously, IP addresses, but there's a lot more metadata. And is this metadata sufficiently protected? Not so much technically, is from a legal perspective. I'll explain this very briefly right now, but it's a hundred pages with 400 footnotes to try to explain all of this. Here's the legal background behind all of this. Here's the technical background, which you, as a lawyer, probably don't know yet. And here are a bunch of case studies which point in different directions.

And here's how to think about this stuff. We have concluded some of the stuff that's being done today is being done wrong from a legal perspective. You know, when we look at the Supreme Court rulings that established this, we can say quite definitively the government does not have the constitutional right to look at this without a search warrant. Why? Because this is what the Supreme Court has said, as viewed through the lens of how the technology works.

I'll give the simplest example is email headers. If you put a BCC on, the email headers are actually part of the email transfer protocol, but the email headers in the message itself are end-to-end. They're not given to a third party, like under the meaning of the law. They're just carried by the third party. And it's a very important legalistic distinction. And the Supreme Court ruled on in 1979.

So, Justice Department says, "You're entitled to the email headers from an email message." And our answer is, "No, we don't think so. And here's why." In terms of how the Supreme Court has ruled, this one has never shown up in court yet, as far as we know, but do we keep watching for it?

Yost: It must be fascinating to be researching these issues that could impact Supreme Court cases in the future and interpreting past ruling and law in terms of changing technology.

Bellovin: Paper's almost done now; I'll submit. Submission window for law papers is early August, early February is. I look back, 100, 120 years, and there were patents issued for today we would call algorithms. This is pre-computers. And the courts still didn't like patenting what they called mental steps, but 100, 120 years ago, there were patents being written this way that were being accepted by the patent office.

Why, and why did it change? I don't know. But I researching this, got stuff from the National Archives and so on, and toss it out there and see what this legal history, in fact, not even law, informs.

Yost: A colleague that I'm working with on this project, Gerardo Con Diaz, wrote a book called *Software Rights* that you might find interesting. It's a history of software patents and copyright, and political economy of IT IP.

Bellovin: Sorry. Software Rights? I will look for it.

Yost: Yes, I really like it, a very insightful book on conceptualizing software, and patenting.

Bellovin: I'm staying away from the policy issues in this paper. This paper saying, here's what went on in the early 20th century, why, and what you know and so on. And none of those patents, though, were ever challenged in the court, so, we don't know what the courts would have said. but patents were very, I showed one of them the first one I found, I showed them to a patent attorney I know, and he said, "My God, the guy was a genius writing a patent like this in 1920."

Yost: So, before we conclude, are there any topics or themes I haven't brought up that you had like to discuss?

Bellovin: Computer security problems are not going away. You know our code base is much higher quality, but the attackers are also much higher quality today and there is much more at risk. I worry about IOT, where I think the big problem is economic. I worry about

cloud computing. I think for a small and medium businesses cloud computing is generally an advantage because they don't have the resources to secure things properly themselves. But when there's a failure, it's going to affect a lot more people. Supply chains, we have become very well educated lately in in what a risk they are.

There's a new focus on liability for security problems. And you know, Biden's National Cybersecurity Policy calls for a role for liability. I have been advocating for this for a very long time. That's one of the things I looked at in the legal chapter in the *Firewalls* book in '94. I think this is a very good thing, but there are problems like what do you do about open-source software?

So, computer security is not going to go away as a problem. I can wave my magic wand and deploy cryptography in two-factor authentication every place. I don't have a big enough magic wand to fix all the software bugs.

Bellovin: Okay.

Yost: Well, thank you so much and congratulations on your incredible accomplishments in your career and so appreciate you sharing your insights today.

Bellovin: Well, thanks. Thank you. Always happy to participate in things like this. As I said, I do history too. Happy you participate.

Yost: I'll do a read-through once it's transcribed and just some very light editing, just correct errors. I see. And then you'll have a chance to edit.

Bellovin: Okay. Sounds good. Catch you later.

Yost: Thanks so much. Have a great day.

Bellovin: Bye. Thanks. Bye.

Yost: Bye.