Authenticated Encryption

Description: After catching up with a comparatively sleepy week of security news, Steve and Leo discuss the need for, and the Internet industry's search for, new standards for "Authenticated Encryption," which simultaneously encrypts messages for privacy while also authenticating them against any active in-flight tampering.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-460.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-460-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We've got the latest security news. And he's going to explain some choices he had to make when implementing SQRL involving Authenticated Encryption. That's next on Security Now!.

Leo Laporte: This is Security Now!, Episode 460, recorded June 17th, 2014: Authenticated Encryption.

It's time for Security Now!, the show that covers your privacy and security online with the man of the hour. You're becoming a hero for security and privacy advocates worldwide, Mr. Steven "Tiberius" Gibson. He's planted the flag.

Steve Gibson: Yeah. I had a server at a restaurant the other day said, "Did you coin the term 'spyware'?" I said, "What?"

Leo: That's a funny thing for a waiter to ask you. That's great.

Steve: Well, yes. I've gotten to know her. And she said, "Guess who was doing some cyberstalking last night?" And I said, "I presume it was you because it wasn't me." She says, "Yes, and guess who was stalked?"

Leo: Wow.

Steve: I said, "Oh, really."

Leo: Is she cute?

Steve: She got - well, yeah. She is. She...

Leo: She's obviously got an eye for Mr. G. I think she likes the moustache.

Steve: She's a dance instructor. Oh, she's completely, like, you know, I keep telling her, just get this - why has this moron not put a ring on your finger yet? Because...

Leo: Oh, she's got a boyfriend.

Steve: Oh, absolutely. She knows who she's going to marry. It's just sort of a matter of formality.

Leo: So I was in Irvine a couple of days ago, as was President Obama, by the way, which made it very difficult to get into and out of Irvine.

Steve: Yeah, in fact, we Irvinites were, like, deliberately working to avoid his caravan. He, like, shut down a whole chunk of the Pacific Coast Highway from Newport down to North Laguna.

Leo: I guess presidents have always done this. But in this post 9/11 era it's worse than ever.

Steve: Yup. And of course he gave the big presentation to UCI's graduates this year. And I chatted with one of my Starbucks gals whose husband is a professor at UCI. And she was a little annoyed because he declined the opportunity to have the whole family go and be present because you had to get on a bus at UCI at 6:00 a.m. in order to start the process of going through security and all the rigmarole for, what, probably a presentation that was in the mid-afternoon.

Leo: Yeah, yeah, I think he spoke at 1:30. So, yeah, but nice part of the country. Lot of sun.

Steve: Lot of sun. We have May Gray and June Gloom are the two terms that we have. Generally the days start off a little, I guess it's like overcast from the ocean, a marine layer, as it's called. And then it generally sort of burns off by midday. And today's just sort of cool and pleasant and very nice. So you were down here?

Leo: I was. Well, I now know why you live there because you're right next to the Santa Ana Airport, which makes it very easy to get in and out. I suspect that's one

of the factors that put you in Irvine. And I know the other one is you're very near Knott's Berry Farm, and that's just - that's where we were.

Steve: Oh, yeah, baby.

Leo: Oh, baby.

Steve: I am a rollercoaster fanatic.

Leo: Well, that is like - I've never seen so many rollercoasters in one amusement park in my life. It's the strangest thing because it's the oldest amusement park in America, was started in the '30s to give people something to do while they waited for their pie. And now it has the Accelerator, which is basically you're shot out of a gun straight up in the air. Of course Lisa went on it and loved it. It's the scariest thing I've ever seen. And they have - that's the most modern. And then you have the same old rickety wood rollercoaster with effectively a coal mine car that you're riding in, rattling down the tracks. They call it the Ghost Rider because it is like out of a ghost town.

Steve: I don't know how it was we were talking about it, but Jenny loves rollercoasters also. And is the Big Dipper in Santa Cruz?

Leo: Santa Cruz. That's very much like the Ghost Rider. The Big Dipper is also a timber - and that's, frankly, it's interesting you should mention it because that is the rollercoaster that soured me on the entire experience when I was in high school. My girlfriend and I went on it.

Steve: For the rest of your life.

Leo: No, literally for the rest of my life. I went on it. It was so terrifying, I fell to my knees on the beach afterwards and said, "Never again. Thank you for saving me, Lord. I will never again, I promise." And I have not been on another rollercoaster since.

Steve: Well, of course the technology...

Leo: Why? How could you like that? You're tricking your body into thinking you're falling to your death. What is pleasant about that?

Steve: Yeah. And the technology has just gone amazing. I mean, you see the commercials, and these kids are, like, strapped into these coffins that are then spun through 360 degrees. It's not just the ride the car over the bumpy trail anymore.

Leo: Right, right.

Steve: I mean, this stuff spins you around and does all kinds of wacky things to you.

Leo: Well, and in a way I think those are safer than the Ghost Rider, which is, like I said, you're basically in a box on a timber, a wood track that's going rattle, rattle, rattle, rattle. Whereas these new ones are very smooth. They actually are holding onto rails.

Steve: Yes, yes. Rubber wheels, all kinds of, like, safety technology.

Leo: You may not even know you're going anywhere on those.

Steve: Remember those Ferris wheels where they put you in a cage, then there was a - you had a ring, and you, like, pulled the ring, and it would lock the cage's axle to the exoskeleton. And then as the Ferris wheel went up, you would no longer be, you know, you would no longer have your bottom be kept down with where gravity was. But it would, like, roll you over. And if you timed it right, you could unlock it at just the right time and swing around and...

Leo: And you're just like a madman. That's what you are.

Steve: I guess I am, yeah.

Leo: This is the Accelerator. You come out, I don't know what the speed is, but it's hydraulic acceleration out of it that pins your head to the back of the seat. And, you know...

Steve: So that initial shock is what takes you all the way up to the top of that.

Leo: All the way through. Not just to the top, the whole thing, you know, the whole ride.

Steve: Right. So basically you get to the top. Now you head down.

Leo: Just barely, yeah. You know, the physics of it must be fascinating. I think that...

Steve: Well, the other thing that's really interesting is it absolutely changes the ride whether you're in the front or the back because, from a physics standpoint, the center of the train is essentially where gravity is having its effect. That is, if you're over a hump,

there's as much in front of you as behind you when you're in the center. So the center feels right with, like, going over a hump. Whereas, if you're on the back, you start accelerating early and essentially get whipped over the hump. And if you're in the front, you are accelerated past the hump and then slow down late. Anyway, obviously I...

Leo: So which do you prefer? The front? Oh, no, aagh.

Steve: Aagh.

Leo: I think I saw your house for just a second.

Steve: It's just a different experience. It just, you know, I like it all.

Leo: I'm sad, but an experience I shall never have. And there it is.

Steve: So, but you were a good sport, and you did go with Lisa.

Leo: I went and watched. And Lisa goes on all of them. She loves them. There's another one where you go up, like, 200 feet, and they just drop you. She loves that stuff. I don't. I just - I don't enjoy it. But I'm glad - so do you go to Knott's Berry Farm? You're literally just miles away.

Steve: No.

Leo: You've never been.

Steve: I also live next to Disneyland, and I don't go there, either.

Leo: Yeah. Well, anyway, I was in your - I should have knocked on your door. I apologize for not doing that.

Steve: Yeah, we have the beach. You know, we have the beach. We have - but mostly I have code. I was coding. So...

Leo: You don't have to be anywhere. You could be anywhere. You just, you know, it doesn't really matter.

Steve: That's true. Jenny has noticed that wherever I am, as long as I have something to read or I can do some work, I'm quite happy there.

Leo: So what is the topic of the day today?

Steve: So we're going to talk about some fundamental technology of the Internet. We haven't done any fundamental Internet technology for a long time. I want to talk about something known as "Authenticated Encryption." We've danced around this in different contexts. The problem is that at first you would think that encrypting something is all you need to do. And that's true if all attackers are passive, that is, if all they can do is view what has been encrypted. Even today's cryptographers agree that modern encryption technology, ciphers, do an astonishingly good job of obfuscating, of giving us privacy.

The problem is we don't, on today's Internet, as we know, have passive attackers. We have active attackers and malicious hackers who are incredibly crafty at figuring out how they can get in there and mess with our technology. And what's surprising is that, as good as encryption is, if all you can do is look at it, it turns out to be very brittle. If you can mess with it, if you can change some bits, if you can be active in that, our encryption technology crumbles with surprising speed. So, again, the smart guys that understood that said, okay, wait a minute. That means encrypting for privacy is not enough. We need to authenticate.

Now, and this is not the kind of authentication we've talked about like with certificates, where we're authenticating the identity of what we're connecting to. This is we're authenticating that nothing has changed in the message since it was sent. So something is done to it at the time it's sent. And we've talked about signatures, cryptographic signatures. And authentication is related to them, but not the same because, if something is signed - and again, remember that the rule is whatever we're doing will be known. These are published standards. They're open. So if we were just saying, oh, we're going to use SHA-1 to sign the message, well, the bad guys would know that. So they'd make some changes and then re-sign it with SHA-1 so that the signature was correct again. So we need something more to apply to the message to protect it from active attack.

And that's our topic. There's some interesting history. There's some sadness and some happiness. And I think it's going to be interesting. So that's our main topic. We have a - and a bunch of obviously - actually a relatively quiet week. So we'll talk about some fun things that happened during the week and then plow into some basic Internet technology I think everyone's going to find fascinating.

Leo: I always enjoy these particular episodes. I think they're so useful. I know a lot of our audience uses them in classrooms and elsewhere. And you could make a classroom of your own. You could learn, if you go through all of our basic technology shows - we should make a DVD of just those, just the basic technology shows. It'd be an education in computer encryption. And you even did how computers work.

Steve: Yeah.

Leo: All right, Steven. Let's at least check a little bit of the security news. I know there's not a whole lot to talk about, but...

Steve: Well, yeah. Not a lot happened, but there was an interesting story that surfaced.

And this has been sort of pending all year. And this is about what happens when a single mining pool of bitcoins acquires too much share of the total network's processing power. Concerns first emerged about five or six months ago, around the beginning of the year, because one particular mining pool operator known as GHash.io was - it was noticed that they had about 40% of the entire mining power within the entire Bitcoin network.

And so of course we've talked about mining pools before, the idea being that, rather than an individual miner just sitting there with his hopefully fast machine, maybe ASIC-driven hashing engine, and with it running 24/7, trying to solve the current Bitcoin problem in order to win a bitcoin - or, let's see, I guess when I won, you got multiple bitcoins for a single hash. I haven't kept track of, like, where we are in the curve. But as we'll remember from our original podcast, the trajectory at which bitcoins get minted is continually slowing down and sort of asymptotically reaching a known number of bitcoins that will ever be minted, at which point it all just now - at that point you just transact them. You no longer mine them freshly.

But the problem with being a solo miner is that you could well spend your whole life, especially these days, when the hashing rate has gone up so high that the Bitcoin network, in the autonomous way it has, has increased the difficulty in order to keep the bitcoin rate following this exact trajectory, that an individual miner has a very low probability of winning a bitcoin. It can happen. But again, it's about statistics because you're just making guesses and then using the hashing engine to check your guess. And if you guess right, then you publish that, and it's agreed upon by other operators within the network, and then you end up winning that coin.

So what's emerged sort of just organically is the idea that individual miners would organize into a mining pool, pooling their resources. And it's a function of the relative mining power that individual participants in the pool have. And the agreement is, if any of us succeed, we will share the winnings prorated based on our processing power. So now in mining pools the bitcoins are being won. And because bitcoins can be fractionated down to such a tiny percentage, all the miners are getting fractions of a bitcoin at a relative rate. And so of course the word spreads. And it's like, hey, would you rather maybe get one in your lifetime, but probably not any? Or would you rather at least get something, even if not a lot, on a more or less kind of cash flow basis? And so mining pooling has, for that reason, come to be very dominant.

And also the other phenomenon that has happened is that there's, as is often the case with the way systems work where the incentives don't prevent it, the guy who's bigger tends to have advantages to help them get even bigger. And so that tends to create an unstable system where you get a monopoly. And what's happened is this one particular mining pool, GHash.io, at the beginning of the year was around 39 to 40% of, that is, all of the mining equipment individually owned and collectively used equaled about 40% of the total hashing power of the Bitcoin network at the beginning of the year.

This week, for a chunk of time, and I've seen a lot of different reports about them coming and going and how long they stayed, but they crossed into 51%. That is, GHash.io got the majority of the processing power, this one mining pool, of the entire Bitcoin network. And, you know, there are maybe like six or seven other major mining pools. And then there's a whole raft of much smaller entities that aren't even - no one's even bothering to track, really.

So the issue is, is this the end of the world? I mean, there are a couple of networked and distributed computing researchers who have a good reputation at Cornell University who run the "Hacking, Distributed" blog. And they first raised the flag around the beginning of the year that this was looking like a problem. And in fact they also came up with the

concept of "selfish mining," where it's possible for miners with a lower percentage to sort of mess up the blockchains of other mining operators.

And this has gotten so complicated. I had a limit to how much time I wanted to spend digging into this because it's gone, like, way beyond the nice theory of Bitcoin that we covered years ago on this podcast when it became clear that Bitcoin was something. And when I looked at the original whitepaper, I thought, this is not only something, this is cool. So I've got links in the show notes for anyone who wants to dig more deeply. But I would say it's controversial whether this really represents a big problem.

Ultimately, I think that the people who say a single miner having the majority of processing power is really a bad idea, they're right. I mean, in principle they're right. The whole point of Bitcoin was it was a decentralized virtual currency. It was the decentralization that was key. Well, arguably, we've lost decentralization. One of the fundamental assumptions behind the way the Bitcoin system protects itself is lost if more than half of the processing power is controlled by a single entity.

So what this means is that it could, not that it would, because the arguments against this being a problem are that it would be in that entity's huge disfavor to destabilize the system or to take advantage of this because there's a lot of auditing technology now in place in the network. And within about 10 minutes, the network collectively would know if there was misbehavior on the part of, for example, GHash.io.

But, for example, these guys who are really, really upset about this, the Cornell University distributed computing guys with the "Hacking, Distributed" blog, they asked themselves rhetorically in their posting last Friday the 13th, they said: "Is this really Armageddon?" And they say: "Yes, it is. GHash is in a position to exercise complete control over which transactions appear on the blockchain and which miners reap mining rewards. They could keep 100% of the mining profits to themselves if they so chose. Some people might say that this is a sensational claim." And actually there are a lot of people who do, who really think that these guys are a little way out in getting overly alarmist. But it's worth, I think, worth understanding their position and considering their point.

They argue: "It's not. The main pillar of the Bitcoin narrative was decentralized trust. That narrative has now collapsed. If you're going to trust GHash, you might as well store an account balance on a GHash server and do away with the rest of Bitcoin." And we'll all save a lot of energy is their argument.

However, there are somewhat less overwrought writers who suggest there are only a couple of things which somebody with a 51% of the network hash rate computational power can do. They could prevent transactions of their choosing from gaining confirmations. I mean, and that's not good. That's not something we want. But they're arguing that it doesn't mean the end of the world. And that would make them invalid, those transactions, potentially preventing people from sending bitcoins between addresses. They could reverse transactions they send during the time they're in control, which would allow double-spend transactions. And they could potentially prevent other miners from finding any blocks for a short period of time.

So there's one researcher, well known in the community, whose name is Peter Todd. He's on a bunch of advisory boards. He's very philosophical. He's got some neat YouTube videos he's put up in some Q&A modes where he clearly demonstrates his understanding of where this has all gone. And he sold, when this happened, half of his Bitcoin holdings, mostly because, despite being evangelical about this, he's an evangelist about Bitcoin, he made a solemn promise to himself based on just sort of the theory of destabilization that,

if this ever happened, he would sell half his holdings. And he's Canadian. And so he said it was in the five figures range, not of bitcoins, I presume, but of probably Canadian currency, he liquidated half. Bitcoin itself did suffer last week as a consequence. It was around $650, which is when Peter sold. It crashed about $100 to about $550. And when I looked last, yesterday, it had climbed back up to 600. So it took a bit of a hit from other people who both knew that Peter was selling and got scared by what this means.

Now, the GHash folks - it's worth noting also that this is an anonymous organization. We don't know who they are. There's something that calls itself, I think it's CXO.io. It's like the entity that runs this. But they've gone public with a press release at this time. And they said - the headline was: "Bitcoin mining pool GHash.io is preventing accumulation of 51% of all hashing power." Now, it's worth noting that they didn't prevent it earlier. But I think they're saying they're going to - they understand the problem. They're going to act to back off. They said: "GHash.io, the world's largest and most powerful mining pool, has entered 2014 with overall hashing power of over 40%, making it the No. 1 pool currently in the Bitcoin network." Of course that's now six months old, seven months old, and no longer true. They have hit 51.

"The pool has gained significant hashing power due to the 0% pool fee, merged mining of alt coins, excellent real-time data presentation, as well as quality 24/7/365 support service." And they said at the time the hashing power of GHash.io consists of 45% of the BitFury ASIC-based mining machines on the network and 55% of the network's independent miners. So this is where people have been flocking. And they finally said: "Although the increase of hash power in the pool is considered to be a good thing, reaching 51% of all hashing power is a serious threat to the Bitcoin community. GHash.io will take all necessary precautions to prevent reaching 51% of all hashing power in order to maintain stability of the Bitcoin network."

So what the powers that be who are, I mean, there's, like, forums full of techies getting together, talking about what to do. And the ultimate solution is known as a "hard fork." The idea would be that a hard fork would be created, meaning that essentially another version of Bitcoin would be created which has new technology, so next-generation technology, specifically designed to deal with the problems which have emerged as this thing has taken off. And it's like it had properties that weren't foreseen. For example, it was probably unforeseeable that there would be this kind of monopoly formation. The presumption was it would maintain its decentralization. And of course this crazy rush to custom hardware, first FPGAs and then full-on custom Application-Specific Integrated Circuits, ASICs, specifically created just to hash at incredible rates in order to win bitcoins as that became useful and worth something.

So the idea is that it looks like it's possible, there are a lot of different ideas that are now in the works for how we fix this. And what would happen is a hard fork would be made where essentially that's the technical term for moving to a next version. And the existing blockchain would be imported into this, and this is the one that would be used going forward without the problems that have been identified and known so far. So just really interesting stuff. I mean, when the world actually gets serious about this, you get side effects that were not really foreseen by Satoshi and the early users of Bitcoin.

Leo: I'm so over Bitcoin. So over it, yeah. I just feel like [raspberry]. By the way, I think now we can step back and talk about that Newsweek cover story. That was completely fallacious, wasn't it. That wasn't Satoshi Nakamoto, that guy they found in your neck of the woods.

Steve: It does sound like it was not the case.

Leo: Haven't heard anything since then. It was like...

Steve: You're right, it just died.

Leo: It's like nobody's going to call Newsweek on this and say, hey, you ready to retract? Unbelievable.

Steve: Yeah, yeah.

Leo: Well, this is why we can't have nice crypto currencies. That's all I'm saying.

Steve: Well, it's a challenge. I mean, and now, of course, you just look, and there's, like, 50 of them. And it's like, what? You know, Dogecoin. There's some Torcoin where I just saw, someone tweeted it the other day, where Tor nodes would receive virtual currency in payment for operating nodes based on how much bandwidth they transited. I mean, there's no end of creativity, of ideas. But I guess we'll just see how this shakes out. I think it's just one more thing that the Internet carries and that will never die. But you're right. It'll leave the limelight, and it will just become something that people do that generates some news every so often, sort of like this.

Leo: Yeah, yeah.

Steve: And sort of like Twitter. There was a really strange thing happened the other day. And I couldn't trace it back all the way to its ultimate genesis. What I heard, what I read in one place was that TweetDeck had had a modification made to allow it to support emojis, you know, the wacky smiley faces and coffee cups and doughnuts and hearts and, you know, the Apple keyboard and I presume the Android has access to these emojis, which are part of the unicode character set.

Leo: They just added 218 new emojis to unicode, didn't they, or something like that.

Steve: Yup, yup. So it's been standardized. And the story goes that an Australian - an Austrian, sorry, an Austrian teenager who goes by the name of Firo, F-I-R-O, was experimenting with TweetDeck, presumably because he knew that it had this new feature, trying to get the service to display the unicode red heart character. And in the process he discovered by accident that doing that, displaying a red heart at the end of a message, turned off or broke or defeated, I don't know what proper verb to use because I don't really understand exactly what happened, but the upshot was cross-site scripting became possible.

That is, we've talked about that a lot in a security context. Browsers read what they show us. That is, it's because they read what is sent from a server that we see anything. So when there's an open angle bracket, you know, or the less-than symbol, then a "B,"

and then the closed angle bracket, or the greater than symbol, that says bold. "B" means bold in HTML. So everything that follows until you cancel the bold is shown in bold. And similarly fonts and tables and pictures and images and buttons, I mean, everything we're used to seeing on browsers is described in HTML. And this HTML has the so-called "tags." And one of them is <script>. It's open angle bracket, s-c-r-i-p-t, and then close bracket. And anything, everything in there, until you end the script, is treated like scripting, like a language that the browser should execute.

So the famous history of cross-site scripting is what happens if a web server or web service ever allows unsanitized text which a user provided to be displayed. And this has been hard because that's what forums are. That's what blogs are. I mean, so much of the evolution of the Internet is user-provided content. Yet it's dangerous. And we've had some fun stories here, Leo, and you'll remember - and in fact even xkcd has done a comic where the school calls [Bobby]'s parents and says, "Your last name is really DROP TABLE?" Because they tried to put [Bobby] DROP TABLE into their school's database and erased it.

Leo: That was the famous "Sanitize Your Inputs" xkcd, yeah.

Steve: Exactly. So anyway, what happened...

Leo: Little [Bobby] Drop Tables.

Steve: So what happened is something about this heart and...

Leo: That's the funniest thing. He just wanted to write a love letter, you know?

Steve: Yeah. And somehow he discovered that it displayed what he wrote. Now, he was a good guy. And so I think he tweeted his discovery, which was his mistake. But then he also tweeted Twitter. And he said, hey, guys, thought you should know, I just discovered that if I put one of your new little hearts at the end of a message, your cross-site scripting filter gets turned off for some reason. So somebody else immediately took advantage of this. They wrote a script which fit in 140 characters, including the probably two bytes for the unicode 16-bit heart, which it was a self-retweeting tweet.

So this little JavaScript would, when it was displayed by TweetDeck, any instance of TweetDeck, TweetDeck would, instead of knowing never to display this content in the script, that broke, so it displayed it. In the process, the browser executed the tweet, and the tweet basically said retweet. So what it looked like is that it was a tweet or a Twitter worm because, with no action required by the user, this thing, when received by TweetDeck, would retweet it to all of the followers of that account. All of those people who were running TweetDeck would, again, with no action, cause it to be retweeted to all of their followers. And so you could see this just geometrically exploded onto the Internet, or across Twitter.

Leo: Well, the good news is not everybody, in fact, not very many people use TweetDeck, so...

Steve: Yes. And so that was - it limited its reach, as far as that went.

Leo: Still, bad.

Steve: Yeah.

Leo: There was no malware involved, though. It just was an involuntary retweet.

Steve: No need to change your account. Don't change your passwords. It was just a tweet that was empowered to retweet itself. And clearly that's not something that Twitter intended.

Leo: No.

Steve: And it's a nice - what I liked was it was such a clean, perfect example of some of the ways that our technology is just, I mean, it wasn't well designed. Yes, thank goodness for scripting because it allows so much more of the web than we would have if we were still delivering the original static, do-nothing web pages of version one of the way the web was put together. Scripting is vastly powerful. But there's a fundamental problem when you're sharing the medium for execution with the message, essentially, with the content. You have to be extremely careful about what you are displaying and what you're executing.

And unfortunately, the way this has all evolved, the code is carried with the content, and this is what happens if we make a mistake. And of course unfortunately all kinds of damage has been caused historically by simple mistakes of this kind. And so we just saw another brief one a couple days ago to remind us, essentially, make sure that doesn't happen. And it had, I mean, those protections were always there. But something about the red heart at the end just sort of said, ah, well, you know, who knows. Don't worry about it.

I got a kick out of another story, and that is, actually, this is directly posted by Microsoft in their Windows Azure blog.

Leo: Azure.

Steve: Azure?

Leo: Azure.

Steve: Azure.

Leo: Just say it simple, Azure.

Steve: And that is that they're now issuing non-U.S. IPv4 addresses for content in the U.S.

Leo: They've run out.

Steve: Yes. And so their blog said: "Some Azure customers may have noticed that, for a VM deployed in a U.S. region, when they launch a localized page on a web browser, it may redirect them to an international site. The following explains why this may be happening." This is, of course, written in Microsoft cautious speak. "IPv4 address space has been fully assigned in the United States, meaning that there is no additional IPv4 address space available. This requires Microsoft to use IPv4 address space available to us globally for the addressing of new services. The result is that we will have to use IPv4 address space assigned to a non-U.S. region" - and I got the idea from reading some examples that this was Brazil - "to address services which may be in a U.S. region. It is not possible to transfer registration because the IP space is allocated to the registration authorities by the Internet Assigned Numbers Authority, the IANA."

And so, anyway, this sort of goes on to explain that they're going to do what they can to work with the various IP geolocation services to update the geolocation tables. Because apparently this arose because people were setting up Microsoft-hosted Azure servers and were being told they weren't in the U.S. because people were geolocating the IP, and it was coming up in other countries. And Microsoft says, uh, yeah, because we ran out of IPv4 space in the U.S. So, yes, still more pressure to move the world to IPv6, when all of these problems go away.

And there's been another credit card breach of unknown size. Brian Krebs picked up on it very quickly. I saw his tweet when - and this was the day before. In fact, he may have been the original source of the news even to the U.S. Secret Service because he spotted thousands of newly stolen cards appearing for sale on one of the underground stolen credit card selling sites which he monitors. And it quickly became clear that these were all related to the P.F. Chang's Chinese restaurant. P.F. Chang's has 204 restaurant branches globally. They're in the U.S., Puerto Rico, Mexico, Canada, Argentinia...

Leo: Argentina.

Steve: Argentina. I don't know what's wrong with me today. Argentinia. I've got too many syllalables. Argentina.

Leo: Yes.

Steve: Chile and the Middle East. Banks have been contacted. I guess it sounds like Brian - apparently what happened is several hundred pages, or the first hundred pages, were made available. He said the advertisement on the online underground shop is being sold under the so-called "Ronald Reagan" batch. That's what it's called. They just give it a name. This is the Ronald Reagan batch of credit cards, which does not list the total number of cards that are for sale, instead just the first hundred pages at approximately 50 cards per page, so 5,000 cards have been published.

And the asking price is in the range from $18 to $140 per card. Brian wrote that many

factors can influence the price of an individual card, such as whether the card is Visa or American Express. Platinum and Business cards tend to fetch higher prices than the Classic or the standard cards and so forth. And I imagine, like, how much time left before the card expires on its expiration date and so forth.

But so banks have been contacted. Using the credit card numbers, they can figure out who the issuers are. And it looks like P.F. Chang's locations in Florida, Maryland, New Jersey, Pennsylvania, Nevada, and North Carolina have been affected. Because P.F. Chang's has no idea what has happened, they have shut down credit card processing throughout their entire organization and have gone back to those fondly remembered mechanical credit card imprinters.

Leo: Wow.

Steve: It's really interesting. They don't know what to trust. They don't know where the breach is. The U.S. Secret Service is involved. But their decision has just been stop processing. And so now you give your credit card to the server, who goes back and goes ka-ching, ka-ching, you know, puts the multipart carbon slips in, and brings that back to you to sign and add your tip to. And then they yank out the middle and give it to you, and they keep the top and the bottom copies. So that's what they're doing now, chain-wide, until they figure out what's happened.

Leo: You know, the real flaw here, and it speaks to the topic of the show today, is that the authentication for these is so weak. Your signature, like that somehow authenticates it.

Steve: Yeah, well, in fact, we know that signature means absolutely nothing. You can just, like, do - and in fact, there have been times when, like, I'm splitting the tab with friends, and it's dark, they'll just say, you know, sign mine, and I'll just do some scrawl.

Leo: Yeah. Nobody checks that. No one cares.

Steve: No, no one knows.

Leo: So, and then they try to use that three-digit number on the back of the card as some form of identification or authentication. But that's - it's weak authentication that's the problem. All you really need is those numbers.

Steve: Well, what's worse is that, while the card will have it physically printed on it, and it's not in the mag stripe, it's like online purchases, you're having to put that in all the time, now, how you...

Leo: All the bad guys have card makers. I mean, I know that because I saw that movie, the Melissa McCarthy, "Identity Thief." They all have the makers that make the thing.

Steve: Yeah, yeah, yeah.

Leo: And the problem with, of course, this paper slip method is now your credit card is completely visible in this stack of papers.

Steve: Right.

Leo: Who's controlling that?

Steve: Right.

Leo: The whole thing's terrible.

Steve: Yeah. Well, and the argument is that chip-and-pin will solve the problem. What chip-and-pin does, I mean, it's an improvement over not having any card-based authentication. But if there was a problem, you'd still fall back to this. I mean, this is still your fallback position. And at some point authentication happens. And after that point, then your data has been authenticated and vulnerable. So we're asking the system to do a lot for us, and it doesn't do it all perfectly.

I wanted to just make a mention in a couple miscellaneous things. I was really very pleased to see that the cover of next week's Time magazine has two words: "Eat Butter."

Leo: That's going to be a bestseller.

Steve: It's got a big shaved piece of butter, kind of curled up like you see in fancy restaurants when they bring sort of like these large butter curls on a plate. And it just says, "Eat Butter." And that's the cover of the June 23rd Time magazine. And the story is "Ending the War on Fat." Because, I mean, boy, things move slowly. I mean, anyone who's been interested in finding the truth has been able to do the research for a long time. Of course I famously, we did the "Over the Sugar Hill" podcast years ago when I stumbled my way into ketoses and thought, what has just happened to me, and did the research to figure out that it's sugar and carbohydrates that are causing all of the problems, and that all of this advice that has been just pounded into us for decades has been completely wrong, and that in fact there's nothing wrong with saturated fat.

And so now they're, like, looking at this again, thinking, well, you know, this whole low-fat, high-carb thing isn't working out too well because the levels of diabetes is just skyrocketing. That was in the news, like, two weeks ago. And it turns out that it is carbohydrate which induces your liver to produce the very low-density liptoproteins which are the easier to oxidize and thus create plaque on arterial endothelial walls. So it turns out it's not eating fat, it's eating something that's absolutely not fat, that's essentially sugar, starch, and that creates arterial fat. So, yeah. Anyway, if anyone's interested, cover of Time magazine next week.

Also, for those who have been watching, the one science fiction series that seems to be hanging on is "Falling Skies." And it begins its fourth season this coming weekend with a

third season marathon. Maybe it even goes back further than third season. I just noticed that it was back-to-back episodes coming up to the premiere of Season 4, so I didn't want anyone to miss it. I'm watching it. You know, none of this stuff is "Star Trek: Next Generation" or "Battlestar Galactica" caliber. We just, you know, those are just so few and far between. But, frankly, of all of the low-budget sci-fi, I'm liking "Falling Skies" enough to have stayed with it. And there's enough eye candy, mechanical and alien and stuff, special effects. It's like, okay, I'll keep watching. And I have to say, Leo, after the third episode, there is no series better named than "Halt and Catch Fire." Oh.

Leo: Terrible?

Steve: It is so awful. It is so bad. It is, you know, I'm sorry I kept mentioning it, hoping that it might be good. And the consensus is generally I get people who tweet me just groaning. It's like, oh. I go, okay. I'll be watching it tonight. I haven't seen it yet. But, yeah. It's not for us. If it was a drama about anything else, I would have never even picked up on it. But it was going to be about computers. Except it's not. I mean, the jargon is misused. It's really not worth even giving any more time of the podcast to because it's just awful. So for what it's worth, for those people who were watching it and thinking, what was Steve thinking, it's like, well, I was just not wanting us to miss it if it was good. And now we know: "Halt and Catch Fire," perfectly named.

Now, I wanted to give a SQRL update because I just spent the last week doing something unexpected. I had an email dialogue with someone who's working on his master's thesis, named Ralf, who Friday before last we had some correspondence. He had some questions - oh, I'm sorry. He's doing his master's thesis on SQRL. So he had some questions. And I've heard from a number of people who are doing similar things, so it's an interesting and popular topic and makes a cool topic for a master's thesis, I think. And he's creating an implementation on Android as part of this.

And so we had some dialogue back and forth. And he asked me, he said, could you put in a switch in the storage format to allow for something other than OCB? OCB is an Authenticated Encryption mode. And I've talked about it before. I've been enamored of it for years because it's the best, and I want to go with the best. The problem is, it's patented. Actually, I think there's, like, four patents covering it. But when I wanted to use it for CryptoLink, which is where this first came up, I wrote to its inventor, Phil Rogaway at UC Davis. He's a cryptographer and has been very involved in Authenticated Encryption. And he said, "Oh, Steve, it was never my intention to profit from little software development firms or people like you. Of course you're free to use it for your cryptographic VPN solution." So I had a free license to use it.

And Ralf liked something called GCM, which I didn't know as much about then as I do now because I spent the week writing an implementation of GCM for public domain, all open source, all cross-platform. It's been compiled on Windows and Mac, on Android, in a bunch of UNIXes, on iOS 7 and 8 beta. And this was a chunk of code that I had to write in "C" because it had to be portable. I'm writing a client just for Windows, so that I could write in assembler, which is where I'm still the most comfortable.

But when I looked around, there was no cross-platform, openly licensed implementation of GCM. They were all GPLed in one flavor or another. And while some people will be creating GPL-licensed clients and server-side code for SQRL, I'm going to publish probably all my source that's useful, but not technically GPLed. And I had a free license. I'm sure that Phillip would have never asked me or had a problem with me using it, which was why I was planning to actually use OCB for SQRL.

But Ralf made the point that this was a patented technology that I was wanting to put in an absolutely free and open public domain protocol where all the other crypto is explicitly public domain. All of Dan Bernstein's curve public key stuff that I'm using, the Scrypt, PBKDF2, I'm trying to think, the Tarsnap guy Colin Percival did, all public domain, free to use. And I just didn't like the idea, after Ralf pointed me at it, that there would be this issue of SQRL's storage container using patented technology. Open as Phillip has tried to make it, there's a GPL license. There's even a commercial but non-military license. Except that, you know, how could somebody who wanted to use SQRL in a commercial setting guarantee that it would never be used by the military?

So anyway, just it got to be a mess. And what was really interesting was, as I was looking at this, just the - so OCB is better. And I'll get into a little bit about why later on when we talk about Authenticated Encryption. But that's why I wanted to use it. It is the best solution. Everybody agrees. Everybody has looked at it. Nobody else has used it because of the patents. And I don't know what the back story is. I know that being in the UC system, I know from personal experience...

Leo: You know exactly, yeah. In fact, that's why I'm surprised that you took the guy's word for it. I mean, doesn't UC have to weigh in, too?

Steve: Yeah, well...

Leo: Don't they have the patent? I mean, maybe he's being a little...

Steve: The way it works is, as I found out when I was at Berkeley, is the Regents of the State of California own the intellectual property created by all of their faculty and students. And when I found out...

Leo: Yeah. So it's nice of him to give you permission, but he may not have had jurisdiction, frankly.

Steve: So what's happened - yes. So as you look over time - because this has been around, Phillip did this, like, 10 years ago. And everyone wants to use it. But nobody does because they're just not sure. So I thought, you know, for SQRL I can't use it, either. Sad as that is. If I ever do CryptoLink, I may go back to it. But some weird things have happened in the meantime, like Intel added an instruction that specifically performs the wacky multiplication that this GCM requires. And so what I really think, when history is written and we look back on OCB, it will be that, despite the fact that it was the best, because there was this uncertainty about a patent, about intellectual property, the industry moved past it. And then people like Intel added instructions to make the solutions that weren't as fast, now faster. And essentially the window of opportunity closed. So...

Leo: Well, in seven years the patent expires, and then you can use it again.

Steve: True, true. But again, I think this is the kind of thing where we will have solved

this problem. And in fact I'm going to be talking a little bit about crypto competitions because there is an ongoing competition now for sort of a "portfolio," they're calling it. They're not going to choose a single cipher the way the AES competition chose Rijndael as the next-generation cipher standard. They recognize more now that one solution doesn't really fit all possible applications, so they're going to spread it around.

So anyway, I spent seven or eight days back in "C," where I haven't been for a long time, creating a portable "C" code. Anyone who's curious, it's already posted up on GRC, although I got some feedback from some guys with other platforms than I had who have, like, added a makefile and added some more platform-independent header stuff. So I will shortly be, probably after the podcast, merging that work and then updating my stuff. But it's all there under Implementation Aids. There's a page in the SQRL site for anyone who's curious.

NIST, the National Institute of Standards and Technology, has published, because GCM, which I chose, which is Galois/Counter Mode, that we'll be talking about a little bit more, because that is an NIST standard, they published a set of six files containing what's known as "test vectors." That is, this is - here's a key. Here's plaintext. Here's authenticated text. What should the answer be? What should the ciphertext be? What should the authentication tag be? That kind of thing. The idea is it's a test suite for validating an implementation.

And so after writing an implementation, which I'm tickled to say is 5K, a little 5K library, because there's lots of tricks could be performed also, I then wrote a validation test using their six files, Perl code that compiles that into a binary, then the binary gets read, and it does 47,250 different tests against my implementation, and of course it passes them all. So we now have in the public domain, license-free for everyone to use, an implementation of GCM. And I'll explain what that is here in a minute.

I got a neat note from Bruce Behrens, who's in Virginia Beach, and he called it a "testimonial/comment." He said: "Hi, Steve. I wanted to let you know that I finally bought a copy of SpinRite. My wife's old Vista desktop was intermittently crashing and giving her the Blue Screen of Death. I decided SpinRite was worth a try." And he said, parents, "(And I've enjoyed the podcast since 2006.) I ran SpinRite, I think on Level 2 first, and then on Level 4, and it didn't report any problems or miraculous recoveries at all. But on the other hand, there has not been a single problem since. My wife suspects a wasted purchase, which wouldn't bother me in the least since, as I said, I've enjoyed the podcast since 2006. But I'm not sure. What do you think? Regards, Bruce."

And Bruce and everyone, we run across this all the time. Unfortunately, not all of what SpinRite does it's able to show because SpinRite works with the drive to help it find problems. And drives are meant to be autonomous. New drives have brains. And so they're intended just to work and to solve all of their problems themselves. Which means, as we were talking about error correction, just to correct errors and not bother us, not even notify us if they're doing that, and to swap out bad sectors when they encounter them and not bother us.

Here we have a situation where that autonomy was failing. The drive was obviously no longer solving its own problems autonomously. Those were surfacing in the form of crashes and blue screens of death; that is, the drive was just - it was dying. And for one reason or another, something about the data that the drive was encountering would cause it to go offline or generate return gibberish, and the system would react by crashing.

So I would think that it was the Level 4 scan, the second one that Bruce did, that

probably did the trick because that one reads the data, inverts it, writes it back, reads it, verifies it, inverts it, writes it back, and reads it again, really giving the drive an exercise and every possible opportunity, essentially, to refresh the storage of the data on the drive. Whereas Level 2 is a very sensitive read pass. It's much faster because it's doing much less work. But it doesn't have the advantage of literally remagnetizing every single bit on the drive, which is Level 4's claim to fame. So that probably basically just - it just sort of went over and just sort of replowed the drive's surface and got all the data settled back down. So I would argue, not a wasted purchase because it did in fact fix the problem, and we probably know why.

Leo: So what is this, this authentication for encryption you're talking about here?

Steve: So we've talked about encryption often. And we've talked about how, for example, a block cipher like the AES standard, which is Rijndael, takes any combination of 128 bits and, under the influence of a key, converts them into a different pattern of 128 bits. And what's so cool is that, if you even just put a counter on the input, if the 128 bits were just a counter, where 0000001, 0000010, 0000011, you know, just binary counting, what comes out despite something so predictable and even barely changing, only a few bits are changing every count, what comes out is cryptographic strength pseudorandom. That is, absolute gibberish, all the bits changing half the time as the counter counts. I mean, absolutely no detectable pattern. And it's keyable. That is, for example, as this counter counts, one sequence comes out, which is completely pseudorandom. But if you change the key, you get an absolutely different sequence coming out. So as a module, as a black box, this cipher is just really neat.

Now, we've also covered on previous podcasts that encrypting something is not as simple as simply taking the so-called plaintext - and in the case of, for example, this Rijndael cipher, which is 128 bits, that's 16 bytes, 16 eight-bit bytes. So you can't simply take your thing to be encrypted in 16-byte chunks and feed each one in and be secure with the output. Every block of 16 will get changed into something pseudorandom. But if you ever put the same block of 16 from, like, later on in the plaintext, it would translate into the same pseudorandom block. And that's not good because that represents some leakage of information. Even though bad guys scrutinizing the so-called ciphertext, which would be the enciphered version of the plaintext, even though they wouldn't instantly know anything, if they found repetitions of 16 bits, I'm sorry, of 16-byte blocks within, they would know there were repetitions in the plaintext. Information is leaking, and that makes cryptographers very uncomfortable.

So the next evolution of this was the addition of so-called modes. So, for example, a mode of encryption is called counter mode, where you do just like I was suggesting in that example. You take a block cipher like AES, and you give it a counter. And in fact you might use an initialization vector as, like, the starting value of this counter so that that changes with each message. And we'll talk about that being important in a second. But the point is that this counter would be encrypted and produce 128 bits output. Then you XOR those with the plaintext to get ciphertext.

And we've talked about the XOR operation before, how it's really interesting. It's also known as "carryless addition" because essentially each bit, bit by bit, is added; but you don't carry from two ones being added which gives you a binary two, which is a one zero. You don't carry that one. All you get, all that's left over is the zero. So the XOR operation is, you can think of it as an OR of either bit, but not both. That is why it's called "exclusive OR," it's exclusive of both, or carryless addition.

But what's interesting about it is, contrary to, like, intuition as it is, if we simply XOR plaintext, where what we're XORing is random, what we get out is unbreakable. Even though it's a simple operation to perform, as long as we're exclusive ORing with something that is a secret, the result is unbreakable. And what's also interesting - and really, think about what we've done. All we've done is we've inverted some of the bits of the input. That's all XOR does is essentially it's a conditional inversion of the input data.

But that's why, when you XOR the ciphertext again with the same random noise, you get back your plaintext because, again, it conditionally reinverts the same bits, and you're back to where you started. So a mode is known as counter mode, where we would take 16-byte blocks of plaintext and successively XOR them with the output of the block cipher as the input is counted. And it simply counts along, and the cipher generates its random noise under the influence of the key, and we XOR our plaintext and get out ciphertext. And that's really good encryption.

Now, one very important caveat is that we never reuse the same sequence of pseudorandom noise with the same key because that breaks the security of this instantly. And as I was saying, the initialization vector might be the starting count of the counter. And so the point is you would want to make sure that you had strong guarantees about the initialization vector and that you not only had different ones, but you didn't have overlapping ranges. So these modes have to be used carefully.

So there's an example of taking a cipher and understanding what the limitations are, and then adding some algorithm to it, these modes, in order to make up for some of the deficiencies of just using the cipher by itself. And I know that anyone who's poked around the 'Net may have seen some famous - there's a famous picture of modeless encryption, that is, making the mistake of not using a mode, where you take the famous Linux penguin, and you encrypt it with like a really good cipher, like AES, with a secret key. And the result is a picture. And even though the color is gone, and it's washed out, you still see the outlines of the Linux penguin. I mean, he survived. So thus information leaked out. And so it's a classic and perfect example of why you can't use a cipher by itself.

Well, as this technology started being looked at more closely and used, cryptographers realized they still had a problem. That is, even using various modes, another common one we've talked about often is cipher block chaining, CBC. That first example, counter mode, is CTR in terms of its three-letter acronym. This was CBC, cipher block chaining. There you take your initialization vector and XOR it with the plaintext. Then you encrypt it with a block cipher, and that gets your cipher, your enciphered code.

And then you take that output from the first block operation, and you use it as the input for the XOR with the plaintext of the second block operation, forming a chain, thus cipher block chaining. And the advantage there is you create a dependency on all of the future cipher operations from the past, and specifically you absolutely break the phenomenon where duplicated blocks of plaintext end up being duplicated in the ciphertext. That won't happen. You'll get no penguin picture. You just get absolute static. Penguin in, absolute static out. That's a very good mode of encryption.

But what researchers discovered was that, if you changed, if an attacker had access to this process, the cipher decryption online, in an online attack, they could still get up to some mischief. And a perfect example is, going back to cipher block chaining, the first thing that is done is the initialization vector is XORed with the plaintext, and then it is encrypted. Well, that means the reverse has to happen for decryption. The ciphertext runs through the block cipher, and then that output is XORed with the initialization vector to get the plaintext. What that means is that, if an attacker changed some bits in the

initialization vector, they're flipping bits in the first block of plaintext. That gives them power.

Even, I mean, if this, for example, this might be a well-known protocol. It might be HTTP or SMTP or a protocol where the format in the header or in the first 16 bytes is fixed or known. And imagine, then, if they know the first 16 characters and could, at will, change them to be exactly anything else they want. They could dramatically change the meaning of the rest of the message. And obviously that can't be allowed to stand. And cipher block chaining happens to be one of the most popular cryptographic modes we have. So the danger is very real.

So if the attack is offline, that is, if there's no way for an attacker to mess with the encrypted data, then there isn't as compelling a case for Authenticated Encryption. But if you are sending encrypted contents online from point A to point B, and there's any chance that a bad guy could modify that data, we have to authenticate it; that is, we have to prevent it from being changed, or rather detect any change, since we would argue we can't prevent it once it's left. It can get changed. The best we can do then is to robustly detect any change.

So the cryptographers got together and thought, okay. We understand the need. What do we do about this? Well, the concept of a Message Authentication Code, a MAC, has been around, had been around for a while. And the technology on MAC composition had been evolving. And so they said, okay, well, this seems pretty simple; right? We want to add a message authentication code to the encrypted data. Then there was some question about, wait a minute, okay, if we have plaintext, and we know we want to do two things to it, we want it to be encrypted and we want it to be authenticated, which one do we do first? Do we authenticate the plaintext and then encrypt all that so that, when we decrypt it, then we authenticate that what's decrypted hasn't been changed? Or do we encrypt it first and then authenticate the encrypted thing and send that?

And so they talked about that for a while. And it turns out it wasn't much competition. The absolute agreement is you encrypt, then you MAC. The argument is that you should never find yourself in a position of being sort of tricked into decrypting data that may have been changed. So if we MAC inside of the encryption, we are going to be decrypting the envelope. And cryptographers don't like that. So the constructions of these authenticated encrypted modes are such that, if the authentication fails, you just stop. You don't, you do not decrypt under any circumstances because it's dangerous. You just - you don't know what the bad guys could have gotten up to. Maybe they found a buffer overflow, god help us, in the decrypter, like in the decryption code. And so they made a modification that, if we were to decrypt it, would give them control of the system. The point is, no, if it is not authentic, stop.

So for a while, that was what we were doing. And when we talk about the various SSL cipher modes and how the names of the modes are formed from AES-128, none are coming to mind, but then SHA-1 or SHA-256 or whatever, this is encryption which is then authenticated. So this concept we've had for quite a while, that we need to protect the contents or the results of the encryption, the contents of the encrypted envelope itself. And if we detect tampering, don't go any further. Absolutely just say, oh, there seems to have been a problem. Could you send that packet to me again? Or whatever is appropriate for the circumstance.

Okay. So let's take a little segue here to talk about the history of cryptographic competitions because that's the way - it's competitions which is the way, sort of the solution that the industry has organically arrived at for coming up with the best standards. Famously, it's where we got Rijndael as the AES standard that has become

now so popular and displaced things like 3DES, which was heavily used in the past. This AES competition is a result of an original announcement made by NIST on January 2nd of '97. And the goal was to come up with a block cipher. They know that it needed 128 bits. We were moving from a world of 68-bit block ciphers, and everyone was beginning to feel uncomfortable. There just weren't enough bits in the block.

Remember that a block cipher simultaneously maps all of its input bits into different output bits using a keyed one-to-one mapping between a combination of input and a combination of output. But the question is, how many combinations there are of input. As drives become ridiculously fast, as hashing engines and custom hardware become ridiculously inexpensive, 64 bits just stops seeming like enough possible combinations. So always being conservative, we doubled it to, I mean, even though 64 was still working, we doubled it to 128 bits, saying that's all we're going to need for quite a while. Keys are different. Keys are inherently ephemeral; and, depending upon their usage, we may need more bits for them. But the block length, due to the security guarantees that the cipher gives us, 128 bits tends to be plenty.

So NIST said we need a 128-bit cipher for safety. We'd like it to have key length of 128 bits, 192, and 256 bits so that we get some future-proofing. We're going to compare it against all of the submissions. All of the challengers will be compared against each other. We want to compare it against how a random function behaves to see how close it is to random because really close would be good. We're going to examine them for their mathematical basis, for any other security factors that may arise as a result of just the general community getting more familiar with this problem. We're going to evaluate them in terms of cost, the licensing requirements.

For example, it needs to be available on a worldwide, nonexclusive, royalty-free basis. Another factor of cost is the computational efficiency. Can it be implemented in hardware? In smartcards? In inexpensive places? So there's not only financial cost but implementation cost. And then the algorithm and implementation characteristics, how flexible is it with key size and block size, and maybe its suitability for other applications, like use as a stream cipher or as a component of a message authentication code, or maybe a pseudorandom number generator, or a hash, as like a core function in a hash. And is it simple? Because it needs to be simple so that people can actually write it and implement it and have it work robustly.

So this was announced in '97, the competition was. All academics from all over the place poured their designs in. And then in a series of deadlines and conferences they broke many. Many were withdrawn. Some were revised and improved. Through several rounds they reduced it. And finally, on October 7th, I'm sorry, October 2nd of the year 2000, NIST announced that the agreement among all of the judges of all of these contestants was Rijndael, which was sort of a funky concatenation of the last names of the designers. And I want to say they were Brazilian. I think they were. Brazil seems to be in the air today.

So since then there have been other competitions which, I mean, this is like Rijndael worked really well, and the industry decided, hey, this concept works. So there is currently an eSTREAM cipher which, well, eSTREAM was - the question was we wanted to come up with a very fast stream cipher as opposed to a block cipher. We've had stream ciphers before. Famously, we've talked about RC4, which was the cipher used in the early version of WiFi. RC4 was incredibly elegant, very efficient, easy to implement in software. It was patented and secret, but the patent did run out. And now everyone knows how to do RC4. Then there is RC5 and 6 and, I think, 8.

Anyway, so these competitions are the way now we are settling on next generations of

these major components that we're using in cryptography. There's the stream cipher competition. There is currently a password hashing competition underway. It's password-hashing.net. And password storage has been a constant topic for this podcast. The site and the organizers of the competition said that: "The password hashing competition is an effort organized to identify new password-hashing schemes in order to improve on the state of the art," which they regard as PBKDF2, Scrypt, et cetera, "and to encourage the use of strong password protection. Applications include, for example, authentication to web services, PIN authentication on mobile devices, key derivation for full-disk encryption, or private key encryption," all the stuff that we talk about. And we talk about how many iterations of a hash a password goes through before it gets stored. We've been talking about this topic through the podcast for years.

So 24 submissions have been accepted of completely brand new password-hashing constructions. In 2013, the first quarter of last year, was the call for submissions. Then in March, at the end of March this year, 2014, was the submission deadline. By the third quarter of this year, the finalists will have been chosen. And the goal is by the second quarter of next year, 2015, there will be the selection made of one or more password-hashing schemes. And they now have 24 submissions at this point that have been accepted. So over the course of the next year, essentially, to about this time next year, they will whittle this down, break them, withdraw them, improve them, and we'll see what we get.

One of the other competitions, which has been started, it was announced at the beginning of last year, January 15th, 2013, is known as the CAESAR competition, C-A-E-S-A-R, which is the acronym or the abbreviation for Competition for Authenticated Encryption: Security, Applicability, and Robustness. So they had to stretch a little bit to get the acronym to work, but CAESAR is the competition that is most new, started beginning of last year. The first deadline round of submissions was March 15th of this year. So the contestants had, what, I guess about 14 months from January 2013 to March 2014. Yeah, 14 months, 57 entrants. A handful have already been withdrawn or modified since.

And their tentative final announcement is, I mean, there'll be many rounds of competition, judging. They know how long this takes. It's just not something that you can do overnight. And this is important. This is to come up with an Authenticated Encryption standard. And again, maybe not a single one, the way we had a single Rijndael cipher chosen for AES. But they're talking in terms of a portfolio, recognizing that because needs vary, some will be applicable in some cases and others elsewhere. And maybe some will be like more strong but slower, where we don't need that much strength today, but it'll be nice to have it thoroughly checked out and ready and maybe even implemented in cipher suites, but not yet deployed because there just isn't that much need for such large blocks or key strength. But anyway, the point is, not till December 15th of 2017. So years from now. Consequently, we can't wait. We will get something in another three and a half years, but nothing until then.

So we're having to move forward. And the industry has moved forward. The difference between the cipher suites that we're used to talking about and Authenticated Encryption - now I'm using too few syllables - is Authenticated Encryption, which is what OCB is, and which is what GCM is, does everything at once. It is not encrypt the plaintext into a plaintext message, then in a second pass create an authentication of it by running the ciphertext through a MAC. The whole point of so-called AE, Authenticated Encryption, is that the encryption is authenticating, too. It is a hybrid mode which is potentially much faster than doing it twice.

And many of the good ones are known as online modes, meaning that you don't need to

know the length of the input in order to compute the output. Meaning you can start feeding the plaintext through a black box. Ciphertext is coming out. And then when you're finally done, and we don't know when that is, you say, okay, this is the end of the message. And then the algorithm completes itself and computes at that time an authentication tag which is a function of everything that came before, very much like a hash. And we've talked about hashing, which is a function of everything that came before except this is under the influence of a key.

And that's another thing that's different about a single hybrid Authenticated Encryption mode is the other ones need separate keys. If you're going to encrypt text and then authenticate it, cryptographers have been very worried about interactions between the two processes, if you use the same key. So the problem with the separate encryption, then authenticate, which is what for example SSL and TLS are typically using now, is they have, for security, you need separate keying material.

The beauty of the hybrid Authenticated Encryption is a single key drives the whole thing. That's one of the things that I really liked about OCB in the beginning. But that is true of many of the other modes, too. Well, in fact, in all of the Authenticated Encryption modes. And many of them are online in the sense that you don't need to know the length of what's coming in order to get going, and it's able to tell, you know, whenever it ends it ends, and that's when you say, okay, we're done, and you compute the authentication tag at the end which is computed under the influence of the key that everything was being done under.

So these AE modes, they've been researched now for about 10 years. And interestingly, it's Phil Rogaway, the inventor of OCB, and patenter, unfortunately, of OCB, who's been in the forefront of Authenticated Encryption. He has designed several other AE modes which actually aren't as good as OCB. Maybe he didn't think they qualified for intellectual property protection, or maybe he didn't think they qualified for intellectual property protection, or maybe they didn't. Or he just thought he had something better up his sleeve. I don't know the back story behind why this OCB got patented except that maybe he had to demonstrate he was inventing stuff for the Regents of the university and so he created some intellectual property and slapped a patent on it in order to prove it.

So there are a bunch of existing AE modes. They vary in their speed, in their complexity, that is, how complex they are to implement, whether they are free or not, whether they have patent encumbrance, how widely they are used now, that is, how well proven. It's becoming very important also to consider whether it can be implemented in hardware because hardware is becoming more soft, and custom hardware is blazingly faster than anything you can do algorithmically in software. That's why, for example, hashing, when there was incredible pressure by Bitcoin to move into hardware, they first went into Field Programmable Gate Arrays and then into fully custom silicon because that's where you get your speed.

But it's certainly possible for an algorithm to be designed to run in software and be just awful to implement in hardware. I mean, you give this to the hardware guys in the lab and say, okay, we need you to turn this into a chip, and they just look at you and say, oh, please, don't make us. Whereas some other equally otherwise good and secure solution might just be a cakewalk in hardware, you know, some big lookup table and wires running around in circles and out comes the answer. So that's one of the other ways that future crypto is being evaluated because we're looking at, for example, the need to encipher speeds that are often many gigabytes per second. Many billions of bytes per second have to be able to be run through a cipher and authenticated on the fly as that fiber optic heads under the ocean. So there really is a need and a consideration on the hardware side.

And then, of course, the other interesting thing is there's this notion of associated data. And this was actually something, again, that Phil Rogaway pioneered. He said, okay, we want to authenticate the ciphertext. But what about if we allowed the authentication envelope to extend further than what was encrypted into what he called "associated data"?

And so, for example, this is very useful, things like the header of the packet. Imagine that you could start your process at the very first byte of the packet, run through all the headers which have to be kept in plaintext in order for the packet to be routed. That's where your IP is. That's where your port number is and your protocol ID and version and other things that describe the contents that you want to encrypt, the encrypted payload of the packet. But you specifically don't want, you can't encrypt that. But you would like it protected. You would like the tag at the very end to include in a cryptographically secure sense the non-encrypted header data so that any hacker who tries to mess with that will similarly break the entire packet. So you're protecting even what isn't encrypted. And that's known as "associated data." And those ciphers are called AEAD, which is Authenticated Encryption with Associated Data.

So, and that was one of the things that I liked about OCB was it did that. But so does the one that the industry has settled on, and that is the so-called Galois, it's spelled G-A-L-O-I-S. And Galois fields are a well-known component of crypto. They're also known as "finite fields." We've talked about that a lot, that this is the idea that you're doing work within a confined bit length. So you're taking bits of a certain length and combining them in some way with maybe bits of the same or other lengths. But you're ultimately doing it within what's known as a finite field, which the short version of that is you only keep the remainder. That is, what you do inherently overflows the bit length, and so you say I don't care about the most significant bits. You keep the least significant bits. You keep the remainder, essentially.

So, for example, in the fact of this Galois counter mode, it uses a finite field of 2^128, which is to say 2^128 - or, I'm sorry, 2^128 is 128 binary bits. And it operates with data that is that size. And there's a place where you're doing a so-called "Galois multiplication," where you're multiplying two 128-bit vectors by, you know, with each other. And we know that multiplication doubles the length of things. Whenever you multiply two things, you're going to get something up to twice as long. So the result of a multiplication of 128 bits with another 128-bit value will be a 256-bit result. This discards the most significant 128 bits, keeping only the least significant, the lower half, essentially.

But there are other modes. EAX, which is another - that is a two-pass mode, also developed by Phillip. Like I said, he's been in this for a long time. But he wasn't involved in GCM. And there are some other modes: CCM, XCBC, IAPN, CWC, and then of course scores more coming from the CAESAR competition. But we don't get that for three and a half years.

So my need in SQRL, and the industry's need for secure communications, where we're under increasingly strenuous performance pressure, we have to minimize overhead. We've talked about the overhead of SSL and encryption and how being able to cache SSL sessions means we're not having to renegotiate SSL sessions all the time, but we are still having to perform encryption and decryption at each end. We want that to be as fast as possible. So we want to drop the need with the current modes which are a round of encryption and then authentication on the outgoing end, and then a round of authentication followed by decryption on the incoming end, thus hybridizing that into Authenticated Encryption.

So GCM is an NIST standard. It's been accepted into OpenSSL. It is in the TLS v1.2 spec. So GCM modes now have assigned numbers, and they're in the standard. Firefox and Chrome supports them both, as I mentioned, as does OpenSSL. And GCM is also available in IPSec, which is the standardized IP security which is part of IPv6 and is available on top of IPv4 now. So that's pretty much the one that has won the game. It is, as the name says, Galois/Counter Mode. The Galois is abbreviated GF, for Galois Field, which is a finite field. That's the authentication portion.

Basically you use what's called a GHASH to hash the initialization vector and the data which has been enciphered. So you're basically doing a hash of the cipher. And the enciphering is a counter. So the counter is initialized with the initialization vector, which it likes to have as 96 bits. If you give it a 96-bit IV, Initialization Vector, it uses those as the top 96 bits of the count. And then the lower 32 bits is a block counter which starts at one and is incremented for every block that you do.

And one of the limitations of this, which is part of the spec and well understood, is since we can't ever allow this to repeat, we cannot encrypt more than 2^32, which is four billion blocks of 128-bit data, which is a lot. But the point being, before you get to that point you just need to rekey. You either rekey, or you come up with a new initialization vector that is a new upper 96 before you allow that lower 32 bits to wrap. Otherwise, that gets fed into the cipher. That gets XORed with your plaintext to create the ciphertext. And that means that decryption is very much the same. That is, you also use AES encryption to produce the same key stream, which you then XOR with the ciphertext to get your plaintext back. That produces the same tag at the end which has been proven secure by the cryptographers that have looked at it. There is no way to spoof this.

So now we have a single pass - oh, very fast, I forgot to mention that in 2010, because this was gaining prominence, Intel added an instruction to their standard x86 and x64 instruction set, which is it performs a quarter of the multiplication. You know you can multiply two things at once, or you can multiply halves of them and then add. It was convenient and made more sense to Intel to do this. So they have a bizarre name instruction which basically it's a 64x64-bit instruction that produces a 128-bit result specifically in support of the GCM mode. And of course we've known for a while they've had the so-called "NI" instructions, which contain a number of special instructions that accelerate AES, that allow AES, the Rijndael cipher, to be accelerated in software.

So we're really seeing the standardized hardware moving towards supporting the standardized protocols. The protocols are becoming open. They are becoming the result of competitions among designers in the academic and commercial world who publish these things freely, who produce open source, open license. All the cryptographers pound on each other's work. Collectively we learn more from that process and whittle down a selection of a few very robust, fundamental kernels of technology which we then use universally on the Internet. And of course it's only after years of that that we find out that there were some problems that no one found, and then we go about fixing those.

So that's Authenticated Encryption. It is the next, sort of the next move forward from the current multiphase encryption. I had to dig into it because I needed it for SQRL because I want to protect any output of SQRL in barcodes or backup codes, and everything has to be encrypted, and we need the authentication in order to verify the one password that the user uses to unlock that in order to verify their identity. So I needed authentication and encryption and wanted to do it the right way. So as I said earlier in the podcast, I changed direction because I decided, even though I had Phillip's permission, SQRL had to be bigger than that. It had to have a universal public domain, completely free, unquestionably open implementation. And now it does.

Leo: Well, that's a relief. Steve Gibson is at GRC.com. That's the place where you'll find SpinRite, the world's finest hard drive maintenance and recovery utility. You'll also find 16Kb versions of this show, transcripts, and of course the forums and a very active community talking about SQRL and other things. If you've got a question, if the Internet allows, we'll have a question-and-answer episode next week, and you can leave those questions at GRC.com/feedback.

We have 64Kb audio and HD as well as SD video of the show available at our site, TWiT.tv/sn and wherever podcasts are aggregated. Or you could use our apps, watch live, because we do it every Tuesday, 1:00 p.m. Pacific, 4:00 p.m. Eastern time, 20:00 UTC on TWiT.tv. But you can also get after-the-fact on-demand audio and video, as always. The apps are great for that, too. Thanks, Steve. We'll catch you next week on Security Now!.

Steve: Thanks, Leo.