# Security Now! #989 - 08-27-24

## Cascading Bloom Filters

### This week on Security Now!

CrowdStrike's president appears in person to accept the "Most Epic Fail" award. A secret backdoor discovered in Chinese-made RFID access key cards. Counterfeit and poorly functioning Cisco brand networking gear in use by major institutions, government and military. A startling SSD performance improvement thanks to SpinRite. When is "Bing" actually "Edge" ... and other errata. Another useful National Public Data breach check service. And what are "Cascading Bloom Filters" and why do they offer the promise of 100% browser local and instantaneous certificate revocation detection?

# Security News

**CrowdStrike Exec's 'Most Epic Fail' Award**

PC Magazine's piece was published on August 12th, two weeks ago, yesterday, but I felt that it was important enough to share and we've had our hands full with other news these past two weeks. PC Mag wrote:

> *CrowdStrike's reputation has taken a beating after last month's massive IT outage. But rather than duck the ridicule, the cybersecurity company decided to "own" the mistake this past weekend* [meaning two weeks ago] *by accepting a cybersecurity humiliation award for "most epic fail."*
>
> *At the DEF CON hacking show, the annual Pwnie Awards recognize achievements and major blunders in the cybersecurity industry. Past "winners" of the Most Epic Fail award include Microsoft, the TSA, and Twitter. This year, there was no question that CrowdStrike would receive the notorious title after the company accidentally distributed a faulty security update that bricked millions of Windows PCs and servers.*
>
> *Although CrowdStrike could have easily ignored the award, CrowdStrike President Michael Sentonas accepted it **in person**, which elicited applause from an audience made up of other cybersecurity professionals.*
>
> *Sentonas said it was important he accept the award so that CrowdStrike could "own" its mistakes. "Definitely not the award to be proud of receiving," he told the audience. "I think the team was surprised when I said straight away that I would come and get it because we got this horribly wrong. We've said that a number of different times. And it's super important to own it."*
>
> *He said he will prominently display the trophy he received at CrowdStrike's headquarters in Texas "because I want every CrowdStriker who comes to work to see it." Sentonas said: "Our goal is to protect people, and we got this wrong. I want to make sure everyone understands these things cannot happen, and that's what this community is about."*
>
> *The gesture received praise from other cybersecurity workers since it's rare for a company to accept the Most Epic Fail award from the Pwnies. Still, CrowdStrike faces a long road to repairing its reputation. A pair of class-action lawsuits have already been filed against the company, demanding it pay damages for causing last month's outage. In addition, Delta Air Lines—which was forced to cancel thousands of flights due to the disruption—is also considering a lawsuit against CrowdStrike and Microsoft.*

I thought it was so cool that CrowdStrike's president went to the trouble and had the class to show up in person and that the attendees of DEF CON also had the good sense to applaud his in person appearance.

**Hardware backdoors discovered in Chinese-made key cards**

I hit upon a piece of news last week that was very disturbing. I didn't have the chance to share it last week because we already had too much to talk about. But everyone needs to hear this. The news has had some coverage but Catalin Cimpanu's Risky Business newsletter had the most comprehensive coverage I've seen. Here's what Catalin wrote:

*A security researcher has discovered secret hardware backdoors in RFID key cards manufactured by a major Chinese company. The backdoors can allow threat actors to clone affected smart cards within minutes and access secure areas. They impact smart cards manufactured by Chinese company Shanghai Fudan Microelectronics that were built using MIFARE Classic chips from NXP. The chips have been on the market since 1994 and have been widely used over the past decades to create smart key cards and access badges in hotels, banks, government buildings, factories, and many other places.*


*The chips have been on the market since 1994 and have been widely used over the past decades to create smart key cards and access badges in hotels, banks, government buildings, factories, and many other places.*

*Because they've been on the market for so long, they also have not escaped the prying eyes of the cybersecurity community, which has previous found several ways to break their encryption and clone MIRAGE-based cards with attacks named Darkside, Nested Authentication, Hardnested, and Static Nested attacks.*

*Over the years, vendors have developed improved versions of their smart cards that shipped with various improvements designed to boost security and prevent some of the discovered attacks. Two of the most popular card designs are the FM11RF08 and FM11RF08S variants—where the S stands for "Security improved version."*

*But in a paper published last week, Quarkslab's Philippe Teuwen says that while researching FM11RF08S cards, he found what proved to be a secret backdoor baked inside Fudan cards. He discovered the backdoor while fuzzing the card's command sets and noticed that the card was answering to undocumented instructions on a specific range.*

*<quote> "Apparently, all FM11RF08S implement a backdoor authentication command with a unique key for the entire production. And we broke it."*

*For the Fudan FM11RF08S cards, that key was A396EFA4E24F. Taking his research further, he also found a similar universal backdoor authentication key for the older FM11RF08 cards, which was A31667A8CEC1.*
*This one also impacted many other Fudan card models, such as the FM11RF32 and FM1208-10. It also impacted card models from Infineon and Philips (renamed NXP), suggesting these companies had likely licensed card technology from the Chinese company.*

*According to Teuwen, the backdoor seems to go back as far as 2007, meaning that many access key cards distributed over the past 17 years can be cloned with physical access within seconds.*

*As it was pointed out in a previous paragraph, RF access cards based on MIFARE Classic were already considered insecure for a long time, but that was due to design errors and not a backdoor. Attackers would still have to waste minutes and physical resources to crack and dump a card's data in order to clone access keys configured on it.*

*An intentional backdoor puts these cards into a whole new threat matrix cell.*

What I find to be stomach-turning about this is that this is an example of the true danger we face when we're outsourcing and importing proprietary microelectronics that cannot be readily audited. And this example is, I believe, crucially important because this is not just a theoretical attack. A theoretical attack is bad enough, like the fact that it **would** be possible to install extra chips onto a motherboard to establish a secret backdoor for a theoretical attacker. But there's nothing theoretical about this. This backdoor capability was actually secretly installed into countless supposedly-secure and supposedly even more secure RFID access cards. This wasn't some mistake, this was deliberate.

We will never know the deeper backstory here. We'll never know why this was done, from how high up in Chinese industry or government the decision was made to subvert an extremely popular family of authentication chip technology. And that really doesn't matter. What matters is that someone said "we'll make super secure chips for you super cheap" and they were trusted not to design-in backdoor access.

We can see now why Apple is being so incredibly circumspect about the servers that they'll be installing into their data centers. At the time that seemed like some really cool over-the-top security precautions. Now it seems not only prescient but quite necessary. We really do have a problem with supply-chain security and the only way it's ever going to get better will be when some means it made to thoroughly audit the design and operation of the technologies we use.

**Counterfeit CISCO networking gear**

In a somewhat related story, BleepingComputer recently carried a piece with the headline: *"CEO charged with sale of counterfeit Cisco devices to govt, health orgs"* and they wrote:

*Onur Aksoy, the CEO of a group of dozens of companies, was indicted for allegedly selling more than $100 million worth of counterfeit Cisco network equipment to customers worldwide, including health, military, and government organizations.*

*According to the criminal complaint, the 38-year-old Florida man ran a massive operation between at least as early as 2013 and 2022, importing tens of thousands of modified low-quality networking devices for as much as 95 to 98 percent off of Cisco's MSRP from Hong Kong and Chinese counterfeiters through a network of at least 19 firms in New Jersey and Florida.*

*The indictment alleges these devices were sold as new and genuine Cisco products through dozens of Amazon and eBay storefronts to customers across the United States and overseas, some ending up on the networks of hospitals, schools, government, and military organizations.*

*The fraudulent Cisco devices sold by Pro Network Entities came with performance, functionality, and safety issues that led to failures and malfunctions, which, in turn, generated significant damages to customers' operations and networks.*

*This happened because the counterfeiters who sold the fraudulent Cisco equipment to Aksoy were modifying older, lower-model products (some previously owned) to make them look like genuine models of new and more expensive Cisco devices.*

*A US Department of Justice (DOJ) press release reads: "As alleged, the Chinese counterfeiters often added pirated Cisco software and unauthorized, low-quality, or unreliable components – including components to circumvent technological measures added by Cisco to the software to check for software license compliance and to authenticate the hardware. Finally, to make the devices appear new, genuine, high-quality, and factory-sealed by Cisco, the Chinese counterfeiters allegedly added counterfeited Cisco labels, stickers, boxes, documentation, packaging, and other materials."*

Wow. In other words, total front to back, soup to nuts counterfeiting. BleepingComputer continues:

*Aksoy's companies (collectively known as Pro Network Entities) generated more than $100 million in revenue, with millions lining the defendant's pockets.*

*However, despite his efforts to fly under the radar by using fake delivery addresses, submitting forged paperwork, and breaking shipments into smaller parcels, between 2014 and 2022, Customs and Border Protection (CBP) agents seized roughly 180 loads of counterfeit Cisco equipment shipped to the Pro Network Entities by co-conspirators in China and Hong Kong.*

*In July 2021, law enforcement agents seized 1,156 counterfeit Cisco devices worth over $7 million after executing a search warrant at Aksoy's warehouse. To top it all off, DOJ says that "between 2014 and 2019, Cisco sent seven letters to Aksoy asking him to cease and desist his trafficking of counterfeit goods." Aksoy allegedly responded to at least two of Cisco's cease and desist letters "by causing his attorney to provide Cisco with forged documents."*

*The defendant was arrested in Miami on June 29, 2022, and was also charged in the District of New Jersey the same day with multiple counts of trafficking counterfeit goods and committing mail and wire fraud. If the charges stand up in court and Aksoy gets sentenced to prove the truth of the allegations, this serves to show how easy it could be to infiltrate critical networks if a threat actor uses a similar approach, selling genuine but backdoored networking equipment.*

So, yep, the bad guys have clearly figured out that we're trusting the supply chain every step of the way and unfortunately there are a great many steps along the way. We know that security requires that every step be secure because it only takes the subversion of one step to break the security of the entire chain.

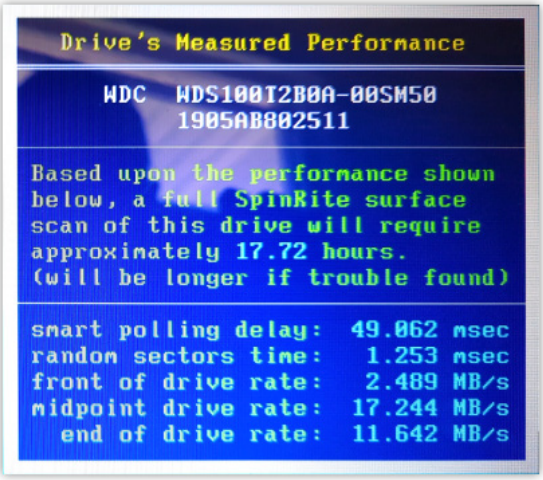
# SpinRite

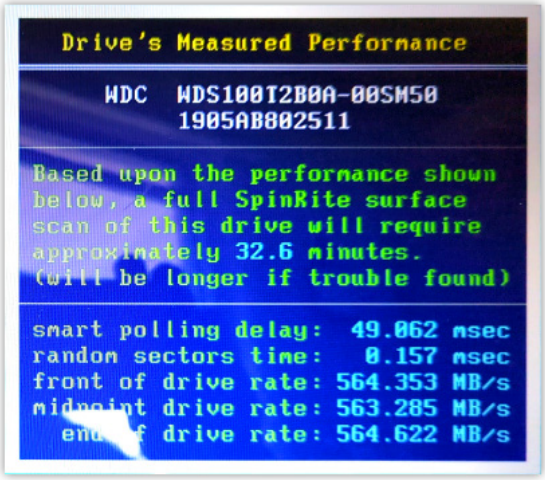
Sandor:

*Thank you for SpinRite. I have a ten year old Toshiba Satellite laptop. I installed a WD Blue SSD just over five years ago – the warranty just expired. This Windows laptop was acting sluggish. I could not immediately identify the issue. Ran down the checklist of items:*

- *Checked for updates and ran checkdisk – no problems.*
- *Checked disk optimization – barely any fragmentation.*
- *Checked the firmware on the SSD – it was up to date.*
- *Ran a quick SMART check – everything was okay.*

*I pulled out SpinRite and ran level 1 and checked my system afterwards. The sluggishness was still there. Thinking about what to check next, I remembered Steve mentioning SpinRite's benchmarking ability. Ran the benchmark and the issue revealed itself – the front, middle, and end reads were way off (2.489, 17.224, 11.642 MB/s); see BenchmarkBefore.jpg. Ran SpinRite at level 3. After level 3 completed, I re-ran the benchmarks which showed how the SSD returned to its proper performance (564.353, 563.285, 564.222 MB/s). See BenchmarkAfter.jpg. The sluggishness I noticed is gone; the Satellite laptop is performing as I expect for a ten year old system.*




# Errata

I have two pieces of errata to share: First, it's been brought to my attention that I've been referring to Microsoft's Edge web browser as "Bing" – which is obviously their illustrious search engine facility. And I'll confess that the two have occupied approximately the same undesirable location in my head. So they were confused. But now that I'm aware of this glitch I'll work to be more conscious of my tendency to lump Edge in with Bing.

The second issue, that was obvious in retrospect, was that the National Public Data breach search facility at pentesters.com requires the choice of a U.S. state. That's not something non-US listeners who were likely affected by the breach – so those in Canada and the UK – have to provide. So in retrospect it's somewhat disappointing that the pentesters site didn't arrange to allow for non-US queries of the dataset.

# Closing the Loop

**Listener Mike wrote:**

*Hi Steve, Just got off the pen test site after doing a name lookup. Yup, I'm there BUT nothing very new. We froze our credit at 4 separate sites many years ago and nothing is showing up since before that happened. We moved to our current house 10 years ago and that address does not show up. Thought you might like to know. Mike*

**And yesterday, listener Kevin White wrote:**

*Hi Steve, Very long time listener. If you count TechTV, I think you and I and Leo go back before SecurityNow existed!*

*Last week you gave us a link to the Pentester site's NPD breach check web page. I just wanted to note that for those of us with a fairly common name, Pentester's check page doesn't always give out complete information.*

*My name is fairly common, and Pentester's check page gave up listing similar names after FirstName "B" LastName ("B" being the middle initial), and my middle initial is in the latter half of the alphabet. So, initially, I thought that I wasn't on the breach list. Then, I checked the https://npdbreach.com/ site against my name and current zip code, and I confirmed I'm on the breach list (sad face).*

*The benefit to the Pentester site is that it appears to show you the full number of entries there are for you on the breach list, whereas the npdbreach site appears to just give you a somewhat distilled version of the results (a single entry for all breach list items with the same SS#, I think).*

*So, it's best to check both sites, especially if you have a common name, or if you have lived in a bunch of different places (checks against SSN not by state).*

*One colloquial data point - a friend was on the list, but it didn't have any recent addresses for him - nothing in the past 15 years. His wife, however, was on the list with their most current address.*

*I guess what's somewhat nice about these circumstances is that the breach list appears to have some data reliability weaknesses that might make it a little more difficult for someone to hijack an updated/recent account. Thanks, and see you tomorrow! -Kevin*

I wanted to make sure that everyone knew about the https://npdbreach.com website. It also appears to be safe and legitimate. They allow the NPD breached data to be retrieved by name and some address info, OR by providing a full social security number OR a phone number. However the site explains that the user's provided social security number or phone number are "blinded" by hashing locally on the browser. They similarly hashed that data at their end and then indexed their database with those hashes. That allows the user to retrieve their records with those hashes without the data being exposed to the site. Not that it makes that much difference anymore since everything is not out in the open for everyone to peruse.

Darren from NSW Australia.
I encountered a really nice note from a listener, Darren, from New South Wales, Australia. I wanted to share it for the perspective that he brings:

*Hi Steve,*

*Long time follower of GRC, short time follower of Security Now. Early last year I was looking for a new podcast when I stumbled across Security Now. Despite being a long time GRC / Steve Gibson follower from the late 90's and visiting your website a lot, particularly in the early days of firewalls for the obligatory Shields Up test on reinstall of Windows, for some reason I never clicked on the Security Now link. When I came across the podcast last year I saw your name and thought to myself "this guy knows what he's talking about, I might have a listen". It was episode #909 "How ESXi fell" and I was hooked. I spend quite a bit of time in the car and I listen to the podcast. So I quickly caught up and was waiting for the podcast for next week when I remembered you and Leo speaking about the fact that GRC has all the previous podcasts available and I thought "Why Not", so I downloaded episodes 1 to 100 and pressed play.*

*What a different world it was back then when you could afford the time to spend a number of weeks telling us how the Internet works and how PKI works as opposed to today when it just seems like there is a never-ending collage of security disasters, vendor mistakes and just plain carelessness. Additionally, I really liked the episodes on NETSTAT (#49) and the history of virtualization (#50). As Leo mentioned recently, compiling your tutorial sessions into a consolidated package would be very valuable for the beginner, which is why I wish I had come across your podcast back in the day as I am an RF Engineer who somehow found myself in the murky world of IT and I am still wondering how that happened. The way you can explain a complex and sometimes abstract concept in simple and understandable terms is an enviable ability. At work, I have to frequently provide a translation service between Engineering and Operations and I have used your examples and explanations a few times to great effect.*

*So my routine is this: I will listen to episodes 1 to 10, then I will listen to the current podcasts, usually two weeks worth given the time it has taken me to listen to 10 episodes, then I will listen to episodes 11 to 20, then the current ones I have missed etc. I am now up to episode #102. I am wondering what my listening schedule will look like as your episodes get longer and longer. I'm thinking I may have to drop back to 5 episodes before tuning in to the current podcasts, otherwise I will be too out of date and I don't want that to happen.*

*I would recommend to all late comers to Security Now to go back to the start and revel in the glory that is Security Now, wonder at how naïve we were back then, and remember a time when a couple of GB of RAM was overkill and way too expensive.*

*Congratulations on a fantastic podcast, a great website, and hopefully we can all be here listening to you and Leo describe the soft landing after the end of Unix time. Thanks and Regards, Darren from New South Wales Australia.*

I also wanted to share this as an example of how different our feedback from email can be from the "here's a link" messaging through Twitter. Don't get me wrong, I know there's a place for both, but the nature of our feedback is completely transformed in a way that works for me. And Darren's point about those older podcasts still being relevant really is true.

# Cascading Bloom Filters

I'm well aware that this podcast's enduring favored feature is pure and clean theoretical deep tech explaining. Leo used to introduce me here as the explainer-in-chief. We were able to do far more of that back in the beginning, starting 19 years ago, before pretty much everything had been thoroughly explained, at least once, and in some cases multiple times. So I'm always on the lookout for something to come along that we have not yet ever explained. Naturally, those events are increasingly rare. But this week we have as pure and clean and deep and theoretical a topic to discuss as we have ever had.

We have listeners who write to tell me that while they do very much enjoy the podcast, and pick up useful tidbits every week, they only, by their own appraisal, feel that they understand around 5% of what is sometimes discussed here. So I'm feeling somewhat self conscious that today's topic is likely to further reduce that percentage. At the same time, there's probably nothing more gratifying than that moment when a new concept is grasped, understood and becomes clear. So I predict that many of our listeners are going to find a lot of new ideas in what follows.

Today's topic is a technique known as a "Bloom Filter." It was so named back on 1970 after its inventor "Burton Howard Bloom." Just to get everyone warmed up here, I'll explain that a Bloom filter is an extremely clever and resource-efficient probabilistic data structure that's used to test whether an element is a member of a set. Although false positive matches are a well known and well understood likelihood, false negatives are not. So in other words, a test using a bloom filter can return either "a datum **may be a member** of the set" or "this datum is definitely **not** a member of the set." Bloom's invention is deliberately information lossy, which is where it obtains its efficiency, but as we'll see, this prevents members from being removed from the set. And the more members that are added to a Bloom Filter, the greater the probability of false positives.

Okay. So why in the world have I suddenly, apparently out of nowhere, decided to talk about Bloom Filters? Where in our recent discussions might it be useful to efficiently know whether something – some item – might be a member of a set of other items? Hmmmm... how about whether a website's certificate might be a member of the set of all currently revoked and non-expired certificates? If we have a really large number of revoked certificates, and we now have web browsers reaching out to scores of other sites, all over certificate-authenticated TLS connections, we need to have some really really fast and really really efficient means for instantly determining whether any of the many TLS certificates the browser receives have been revoked. And it just so happens that Burton Howard Bloom's 54-year old invention from 1970 is the best way to do that today in 2024.

One additional point is that today's podcast is not just titled "Bloom Filters". It's titled "Cascading Bloom Filters". The reason for this, as we'll see, is that the use of a cascade of successive Bloom filters incredibly elegantly completely solves the false positive problem that's inherently created by the Bloom filter's efficiency.

And it turns out that this Bloom Filter technology can come in very handy. The servers of the massive content delivery network Akamai, use Bloom filters to prevent the so-called "one-hit-wonders" from wasting their cache space. One-hit-wonders are web objects requested by users

only once. After Akamai discovered that these one-hit-wonders were needlessly tying up nearly three-quarters of their caching infrastructure, they placed Bloom Filters in front of their cache to detect a second request for the same object. Only if an object was requested second time would they bother to save that in their cache. Google's Bigtable, Apache HBase & Cassandra, ScyllaDB and Post-greS-Q-L all employ Bloom filters to bypass disk lookups for non-existent rows or columns, because quickly knowing what's **not** in a database significantly increases the system's overall performance. For a while, Google's Chrome browser used a Bloom filter to identify potentially malicious URLs. Every URL was first checked against a local Bloom filter, and only if the filter returned a positive result was a full check of the URL then made. Microsoft's Bing (and I do mean Bing and not Edge!) uses multi-level hierarchical Bloom filters for its search index known as BitFunnel. Even Bitcoin once used Bloom filters to speed up wallet synchronization until privacy vulnerabilities with the implementation were discovered (though it wasn't the Filter's fault.) Even the website Medium uses Bloom filters to avoid recommending articles a user has previously read. And Ethereum uses Bloom filters to quickly locate logs on the Ethereum blockchain.

We don't really care about Akamai, Bigtable, HBase, Cassandra, Post-gres-Q-L, Bing, Bitcoin, Ethereum or Medium. But we do care about Mozilla and Firefox and about the fact that Firefox has always appeared to be leading the pack when it comes to actually protecting their users from revoked certificates. As we've all now seen, the other browsers, most notably Chrome, just pays empty lip-service to revocation. So what do we know about Mozilla's plans for next generation CRL – Certificate Revocation List – based revocation?

https://blog.mozilla.org/security/2020/01/09/crlite-part-1-all-web-pki-revocations-compressed/

Four and a half years ago, in January of 2020, Mozilla's blob posting was titled: "Introducing CRLite: All of the Web PKI's revocations, compressed" and they wrote:

> *CRLite is a technology proposed by a group of researchers at the IEEE Symposium on Security and Privacy 2017 that compresses revocation information so effectively that 300 megabytes of revocation data can become 1 megabyte. It accomplishes this by combining Certificate Transparency data and Internet scan results with* ***cascading Bloom filters****, building a data structure that is reliable, easy to verify, and easy to update.*
>
> *Since December,* [so that would have been 2019] *Firefox Nightly has been shipping with CRLite, collecting telemetry on its effectiveness and speed. As can be imagined, replacing a network round-trip with local lookups makes for a substantial performance improvement. Mozilla currently updates the CRLite dataset four times per day, although not all updates are currently delivered to clients.*

Since the CA/Browser forum's members have almost unanimously voted to drop use of OCSP in favor of requiring a return to the original Certificate Revocation List approach, before we dig into what Bloom filters are and how they work, let's take just a moment to understand Mozilla's position on OCSP and CRLs. In that January 2020 posting, they wrote:

> *The design of the Web's Public Key Infrastructure (PKI) included the idea that website*

*certificates would be revocable to indicate that they are no longer safe to trust: perhaps because the server they were used on was being decommissioned, or there had been a security incident. In practice, this has been more of an aspiration, as the imagined mechanisms showed their shortcomings:*

- *Certificate Revocation Lists (CRLs) quickly became large, and contained mostly irrelevant data, so web browsers stopped downloading them;*

- *The Online Certificate Status Protocol (OCSP) was unreliable, so web browsers had to assume if it didn't work that the website was still valid.*

*Since revocation is still crucial for protecting users, browsers built administratively-managed, centralized revocation lists: Firefox's OneCRL, combined with Safe Browsing's URL-specific warnings, provide the tools needed to handle major security incidents, but opinions differ on what to do about finer-grained revocation needs and the role of OCSP.*

*Much has been written on the subject of OCSP reliability, and while reliability has definitely improved in recent years (per Firefox telemetry; failure rate), it still suffers under less-than-perfect network conditions: even among our Beta population, which historically has above-average connectivity,* ***over 7% of OCSP checks time out today.***

*Because of this, it's impractical to require OCSP to succeed for a connection to be secure, and in turn, an adversarial monster-in-the-middle (MITM) can simply block OCSP to achieve their ends. Mozilla has been making improvements in this realm for some time, implementing OCSP Must-Staple, which was designed as a solution to this problem, while continuing to use online status checks whenever a server fails to staple a response.*

*We've also made Firefox bypass revocation information for short-lived certificates; however, despite improvements in automation, such short-lived certificates still make up a very small portion of the Web PKI, because the majority of certificates are long-lived.*

I should mention that when they talk about "short lived" they're talking about hours or days, never months. Thus, they consider Let's Encrypt's 90-day certs to be long-lived. And what Mozilla says next is quite interesting. It's something we've never considered. They wrote:

*The ideal in question, is whether a Certificate Authority's (CA) revocation should be directly relied upon by end-users.* [ ...What? ] *There are legitimate concerns that respecting CA revocations could be a path to enabling CAs to censor websites. This would be particularly troubling in the event of increased consolidation in the CA market. However, at present, if one CA were to engage in censorship, the website operator could go to a different CA.*

*If censorship concerns do bear out, then Mozilla has the option to use its root store policy to influence the situation in accordance with our manifesto.*

That's quite interesting. They must not be talking about typical certificate authorities in the US, but perhaps CA's located in repressive governments which might be forced to revoke the certificates of websites whose certificates had been issued by them which the government no longer approves of.

And then I assume that since Mozilla would certainly disapprove of CA's being used as censorship enforcement, that's where Mozilla's root store policy would come in, with them completely withdrawing all trust from any such CA's signatures, much as the industry recently did with Entrust. Again, they continue:

> *Legitimate revocations are either done by the issuing CA because of a security incident or policy violation, or they are done on behalf of the certificate's owner, for their own purposes. The intention becomes codified to render the certificate unusable, perhaps due to key compromise or service provider change, or as was done in the wake of Heartbleed.*
>
> *Choosing specific revocations to honor, while refusing others dismisses the intentions of all left-behind revocations attempts. For Mozilla, it violates principle 6 of our manifesto, limiting participation in the Web PKI's security model.*

Switching from issues of policy back to issues of technology, they note:

> *There is a cost to supporting all revocations.*
>
> - *Checking OCSP slows down our first connection by ~130 milliseconds*
> - *Fails open, if an adversary is in control of the web connection, and*
> - *Periodically reveals to the CA the HTTPS web host that a user is visiting.*
>
> *Luckily, CRLite gives us the ability to deliver all the revocation knowledge needed to replace OCSP, and do so quickly, compactly, and accurately.*
>
> *Can CRLite Replace OCSP? Firefox Nightly users are currently only using CRLite for telemetry, but by changing the preference security.pki.crlite_mode to '2', CRLite can enter "enforcing" mode and respect CRLite revocations for eligible websites. There's not yet a mode to disable OCSP; there'll be more on that in subsequent posts.*

We can broadly classify major cryptographic algorithms into two classes, lossy and non-lossy. All encryption is non-lossy, meaning that encrypting a source plaintext into a cyphertext does not lose any of the original's information because decryption perfectly reverses the encryption process to restore the original text.

By contrast, cryptographic hashing is a deliberately lossy process. And for what a hash provides that's what we want. As we know, a hash, also known as a digest, takes an original plaintext of any size and reduces it to a single fixed-length output consisting of some number of bits. The number of bits is chosen to be high enough so that the probability of two different texts hashing to exactly the same combination of every bit is astronomically small.

Common sense tells us that it would not be possible to take the entire text of War & Peace, hash it into a tiny 256-bit digest, then somehow reverse that process to get the original text back. "War & Peace" is definitely not all there in those 256 bits even though every character of War & Peace has an effect upon the hash's final output.

Now let's look at the design of Bloom Filters, an entirely different system which, like a hash, is similarly deliberately lossy.

A bloom filter starts with a large array of binary bits all set to zero. The optimal size of the large array is determined by many interacting factors but for our purpose we're going to use an array containing 1,048,576 bits. Since that's 128 Kbytes, that's a workable size today.

Now imagine that we want to "add" an awareness of a revoked certificate to the Bloom filter, to this one million bits which are all initially set to zero. We need to use some fixed characteristic of the revoked certificate. All certificates contain a thumbprint. It was traditionally generated by taking the SHA1 hash of the entire certificate and later that's been upgraded to SHA256. So older certificates have 20-byte or 160-bit thumbprints and newer certificates have 32-bytes or 256-bit thumbprints.

One of the cool things about a cryptographically strong hash function like SHA1 or SHA256 which we've talked about before, is that all of the bits in the hash are equal. A smaller hash can be taken simply by using some subset of the hash's entire value, and it doesn't matter which smaller subset is chosen. So for our example, we're going to take the lowest 20 bits of the revoked certificate's thumbprint hash. Thanks to the thumbprint being a hash, these are effectively pseudo-randomly set bits. Because 2 raised to the power of 20 is 1,048,576, the lowest 20 bits of a thumbprint can be used as the index into our bit array. In other words, it can be used to select exactly one bit from our one megabit array. And we set that single chosen bit to a '1'.

Now let's say that we add more, many more, revoked certificates to this blossoming Bloom filter. Let's say that we add a total of 100,000 revoked certificates. Since adding each certificate entails taking the lowest 20 bits of its pseudo-random but fixed thumbprint and setting the corresponding bit in the bit array to a '1', this would mean that up to 100,000 bits out of our total of one million would have been set after we added those 100,000 revoked certificates.

I say "up to 100,000", though, because there's a problem. There would be a good chance that out of those 100,000 certificates, many of the certificates would share the same lower 20 bits. The other surprising fact is that the number of these collisions turns out to be much higher than we might intuitively guess. We've talked about "the birthday paradox" which explains why the probability that any two certificates might share the same lower 20 bits is much higher than we would think. In practice this means that fewer than 100,000 sets will have been set to '1' due to collisions where the bit that was set to '1' had already been set to '1' by the addition of an earlier certificate.

Also, note that after adding 100,000 certificates, less than 10% of our total bit space has been used. Bloom realized that this was inefficient. And he had a solution: Instead of only setting 1 single bit per certificate, let's set a several. And we can do that easily by taking successive 20-bit pieces of the certificate's thumbprint and using each successive 20-bit piece to set an additional bit. So let's set five bits in the one megabyte array for every certificate that we add to it.

So we start with an empty array with all of its 1,048,576 bits set to zero. We add the first certificate by taking each of five chunks of 20-bits from its thumbprint and using each chunk to determine which bit to set. After adding that one certificate we'll almost certainly have five bits set. It could be that a certificate might collide with itself, though the chances are slight. And then as before, after finally adding all of our 100,000 certificates, with each certificate setting its five

very specific bits to '1', we'll have somewhere less than half of the array's bits set due to collisions... but we'll be very nearly at 50% '1' bits set.

Now let's say that our web browser receives a certificate it wishes to check. It wants to ask the question "has this certificate previously been added to this bloom filter?" To answer that question we take the same five sets of 20 bits from the candidate certificate's thumbprint and we successively use each set of 20 bits to TEST each of those five bits in the bloom filter. Here's what we learn from that, and think about this: If **any** of the five tested bits are '0' then we absolutely positively know that the certificate in question could never have been previously added to the Bloom filter's bit array. Because if it had been then we know that **all** of its bits would be '1's. So if any of the five bits are '0', this certificate is definitely not revoked.

But the problem we still have, if all of the certificate's tested bits are '1's, is that we cannot be certain that this is not a good certificate whose five chosen bits just happen to collide with bits that were set by any of the other revoked certificates. Remember that I said earlier that *"Although false positive matches are a well known and well understood likelihood, false negatives are not. So in other words, a test using a bloom filter can return either "a datum **may be a member** of the set" or "this datum is definitely **not** a member of the set."* The "definitely not a member of the set" is what we get when at least one of the five bits are '0' since that can only happen if this certificate was never added to the filter.

So what do we do about the big problem of false positives created by a Bloom filter? First, it's worth noting that many applications of Bloom filters are not sensitive to false positives. If Akamai occasionally caches a "one-hit-wonder" file because the file's Bloom bits happen to collide with another file, who cares? An extra file gets cached, big deal. Or if Medium is using individual per-user bloom filters to avoid having users see the same article twice, but that occasionally misfires and shows them something they saw before, who cares? Or if a database mistakenly believes that a record exists and goes to look for it instead of immediately rejecting the request thanks to a Bloom filter, again, who cares? So before we solve the problem of Bloom filter false positives, I want to clearly make the point that a simple Bloom filter that is able to almost instantly make a go / no-go determination which never generates a false negative but occasionally generates false positives, can still be an extremely useful tool in computer science.

There are several important points which are worth pausing to highlight here: The first is, that in return for the Bloom filter's discrimination "fuzziness" we get amazing efficiency. The hashes of 100,000 certificate thumbprints can be sorted, but they cannot be compressed because thumbprints are, by definition, incompressible pseudorandom data. Each modern SHA256 thumbprint is 32 bytes. So any simple list of those 100,000 revoked certificate thumbprints would require 3.2 megabytes just to store the list. Yet our trained Bloom filter requires only 128K. So in return for some inaccuracy we obtain a factor of just below 25 times compression.

And then there's speed. Searching a 100,000 entry sorted list is time consuming. And more sophisticated tree structures consume more space and still take time to traverse. By comparison, consider our Bloom filter. The certificate we want to test already offers its thumbprint. We need to perform five tests, each one which directly tests a single bit of the Bloom array. For those who are not coders, any coder will tell you that directly addressing and testing a bit in an array is one of the fastest things a computer can do. Intel's instruction set has a single instruction that does

exactly that, instantly. So nothing could be faster than these test. Now consider that with Bloom array having fewer than half of its bits set, each of these tests has a better than 50/50 chance of selecting a '0' – and the instant we hit **any** '0' in the array we know that this certificate being tested cannot be represented within this array. So with five tests, each super-fast and each having a better than even chance of giving us an immediate go ahead, this has got to be the fastest, most efficient and economical way of classifying a certificate as being a member of a larger set.

**But...** Unlike Akamai, Medium and the databases that can all tolerate false positives, this application of Bloom filters for certificate revocation absolutely cannot tolerate any false positives. We can never tell a user that a remote server has just served up a revoked certificate when it hasn't. So now what do we do?

You get extra credit if you guessed: "This is where the "Cascade" enters the picture."

To pull this off it is necessary for the creator of the final CRLite Bloom filter cascade to have access to the ENTIRE real time active certificate corpus. In other words, a record of every single unexpired certificate, both valid and revoked. Although this might appear to be a tall, if not impossible, order, the PKI public key infrastructure actually already has all of this in place. All CA's are required to publish every certificate they issue into a Certificate Transparency log. And that log exists. So check this out:

A first and largest Bloom filter bit array is first populated by adding all known revoked certificates. As we've seen, this will generate an array with lots of '1' bits set. And we know that once this has been done, any certificate that has any '0' bit CANNOT be a member of the set of all revoked certificates. We also know that each of the five tests has a better than even chance of returning a '0', so most good certificates will be identified almost instantly. But, for this application we must enforce a zero-tolerance policy. So get a load of this! ...

Next, every non-revoked known valid certificate in existence is run through this first stage Bloom filter to explicitly detect every single false positive that the first level Bloom filter produces. And what do you think we do with those? That's right, those false positives are used to train the second-level filter in the Bloom filter cascade.

Therefore, any certificates that are present in the second level cascade are known to actually be valid because they were falsely flagged by the first level when valid certificates were tested. This second level cascade can be thought of as a whitelist to handle the false positives which are generated by the first level cascade. Thus, any certificate that is not in the first level must be valid. We know that. That's the immediate okay.

Then, any certificate that WAS identified by the first level (which we might call a candidate revoked) and is NOT also in the second level MUST actually be revoked since the second level was trained on **valid** certificates that were thought to be bad by the first level. We know that Bloom filters false positive, but they never false negative. So any certificate that the first first level finds which the second level does NOT find must not have participated in the second level's valid certificate training... so we absolutely know that it had to have been revoked.

However, there's one last bit of mind-bending logic to cover. Since Bloom filters are known to produce false positive matches, there's a chance that the second level Bloom filter might produce a false positive. That would lead it to believe that a revoked certificate that was found by the first level filter had been found to be good by the second level filter. So, believe it or not, a third level filter is used to catch and categorize those cases that may slip through the second level filter. But that third level filter is the final stage of the Bloom filter cascade and its decision is final.

So, if the first level filter does NOT produce a match we immediately know that the certificate MUST be valid since that first level was trained on revoked certs and Bloom filters never false negative.

But if that first level filter DOES produce a match, we test the certificate against the second level filter. If it does NOT produce a match, we absolutely know that the certificate MUST have been revoked since the second level filter was trained on valid certificates and, again, Bloom filters never false negative.

And finally, since the third level filter is again trained on revoked certificates, if it does not find a match we absolutely know that the certificate is good and if it does find a match, thanks to the pre-weeding out of the earlier filters, we can finally know with certainty that the certificate is a member of the revoked set and must not be trusted.

I should also note that since the second level Bloom filter only needs to be trained on the VALID certificates that mistakenly false positive match at the first level, that total number of certificates is very much smaller, so a much smaller bit array and filter will suffice. And that's even more true for the third level. So we obtain significantly greater efficiency with successive levels of the cascade.

Although this can all be a bit mind-bending, what we have now is a way of constructing an efficient means of rapidly categorizing certificates as valid or revoked. In order to do this we need to train a trio of Bloom filters on every possible valid and revoked certificate that they might ever encounter in the wild, and that's actually not impossible. Once this has been done, and Mozilla has been doing this now for several years, we wind up with a small local certificate classifier that every user can efficiently run in their own browser. And it can instantly identify any revoked certificate from among millions of candidates without ever making a single error.

This amazing local browser capability is supported and made possible by a massive, continuous, behind the scenes effort to produce and distill, four times per day, those three simple Bloom filter bitmaps which are then made available to every browser over a common content delivery network. Individual Firefox browsers reach out to obtain the latest set of bitmaps every six hours. If or when all browsers have access to these filters we'll finally have solved the certificate revocation problem with both low-latency and total privacy, since ever browser will then be empowered to make these revocation determinations locally.