

Shamir Secret Sharing

It's 3am. Paul, the head of PayPal database administration carefully enters his elaborate passphrase at a keyboard in a darkened cubicle of 1840 Embarcadero Road in East Palo Alto, for the fifth time. He hits Return. The green-on-black console window instantly displays one line of text: "Sorry, one or more wrong passphrases. Can't reconstruct the key. Goodbye."

There is nerd pandemonium all around us. James, our recently promoted VP of Engineering, just climbed the desk at a nearby cubicle, screaming: "Guys, if we can't get this key the right way, we gotta start brute-forcing it ASAP!" It's gallows humor – he knows very well that brute-forcing such a key will take millions of years, and it's already 6am on the East Coast – the first of many "Why is PayPal down today?" articles is undoubtedly going to hit CNET shortly. Our single-story cubicle-maze office is buzzing with nervous activity of PayPalians who know they can't help but want to do something anyway. I poke my head up above the cubicle wall to catch a glimpse of someone trying to stay inside a giant otherwise empty recycling bin on wheels while a couple of Senior Software Engineers are attempting to accelerate the bin up to dangerous speeds in the front lobby. I lower my head and try to stay focused. "Let's try it again, this time with three different people" is the best idea I can come up with, even though I am quite sure it will not work.

It doesn't.

The key in question decrypts PayPal's master payment credential table – also known as the giant store of credit card and bank account numbers. Without access to payment credentials, PayPal doesn't really have a business per se, seeing how we are supposed to facilitate payments, and that's really hard to do if we no longer have access to the 100+ million credit card numbers our users added over the last year of insane growth.

This is the story of a catastrophic software bug I briefly introduced into the PayPal codebase that almost cost us the company (or so it seemed, in the moment.) I've told this story a handful of times, always swearing the listeners to secrecy, and surprisingly it does not appear to have ever been written down before. 20+ years since the incident, it now appears instructive and a little funny, rather than merely extremely embarrassing.

Before we get back to that fateful night, we have to go back another decade. In the summer of 1991, my family and I moved to Chicago from Kyiv, Ukraine. While we had just a few hundred dollars between the five of us, we did have one secret advantage: science fiction fans.

My dad was a highly active member of *Zoryaniy Shlyah* – Kyiv's possibly first (and possibly only, at the time) sci-fi fan club – the name means "Star Trek" in Ukrainian, unsurprisingly. He translated some Stanislaw Lem (of *Solaris* and *Futurological Congress* fame) from Polish to Russian in the early 80s and was generally considered a *coryphaeus* at ZSh.

While USSR was more or less informationally isolated behind the digital Iron Curtain until the late '80s, by 1990 or so, things like FidoNet wriggled their way into the Soviet computing world, and some members of ZSh were now exchanging electronic mail with sci-fi fans of the free world.

The vaguely exotic news of two Soviet refugee sci-fi fans arriving in Chicago was transmitted to the local fandom before we had even boarded the PanAm flight that took us across the Atlantic [1]. My dad (and I, by extension) was soon adopted by some kind Chicago science



too long to tweet

Max Levchin: bilingual bicyclist, eternal entrepreneur.

fiction geeks, a few of whom became close friends over the years, though that's a story for another time.

A year or so after the move to Chicago, our new sci-fi friends invited my dad to a birthday party for a rising star of the local fandom, one Bruce Schneier. We certainly did not know Bruce or really anyone at the party, but it promised good food, friendly people, and probably *filk*. My role was to translate, as my dad spoke limited English at the time.

I had fallen desperately in love with secret codes and cryptography about a year before we left Ukraine. Walking into Bruce's library during the house tour (this was a couple years before *Applied Cryptography* was published and he must have been deep in research) felt like walking into Narnia.

I promptly abandoned my dad to fend for himself as far as small talk and canapés were concerned, and proceeded to make a complete ass out of myself by brazenly asking the host for a few sheets of paper and a pencil. Having been obliged, I pulled a half dozen cryptography books from the shelves and went to work trying to copy down some answers to a few long-held questions on the library floor. After about two hours of scribbling alone like a man possessed, I ran out of paper and decided to temporarily rejoin the party.

On the living room table, Bruce had stacks of copies of his fanzine Ramblings. Thinking I could use the blank sides of the pages to take more notes, I grabbed a printout and was about to quietly return to copying the original S-box values for DES when my dad spotted me from across the room and demanded I help him socialize. The party wrapped soon, and our friends drove us home.

The printout I grabbed was *not* a Ramblings issue. It was a short essay by Bruce titled *Sharing Secrets Among Friends*, essentially a humorous explanation of Shamir Secret Sharing.

Say you want to make sure that something really really important and secret (a nuclear weapon launch code, a database encryption key, etc) cannot be known or used by a single (friendly) actor, but becomes available, if at least n people from a group of m choose to do it. Think two on-duty officers (from a cadre of say 5) turning keys together to get ready for a nuke launch.

The idea (proposed by Adi Shamir – the **S** of RSA! – in 1979) is as simple as it is beautiful.

Let's call the secret we are trying to split among m people K .

First, create a totally random polynomial that looks like: $y(x) = C_0 * x^{(n-1)} + C_1 * x^{(n-2)} + C_2 * x^{(n-3)} \dots + K$. "Create" here just means generate random coefficients C . Now, for every person in your trusted group of m , evaluate the polynomial for some randomly chosen X_m and hand them their corresponding (X_m, Y_m) each.

If we have n of these points together, we can use Lagrange interpolating polynomial to reconstruct the coefficients – and evaluate the original polynomial at $x=0$, which conveniently gives us $y(0) = K$, the secret. Beautiful. I still had the printout with me, years later, in Palo Alto.

It should come as no surprise that during my time as CTO PayPal engineering had an absolute obsession with security. No firewall was one too many, no multi-factor authentication scheme too onerous, etc. *Anything* that was worth anything at all was encrypted at rest.

To decrypt, a service would get the needed data from its database table, transmit it to a special service named *cryptoserv* (an original SUN hardware running Solaris sitting on its own, especially tightly locked-down network) and a special service running only there would perform the decryption and send back the result.

Decryption *request rate* was monitored externally and on *cryptoserv*, and if there were too many requests, the whole thing was to shut down and purge any sensitive data and keys from

its memory until manually restarted.

It was this manual restart that gnawed at me. At launch, a bunch of configuration files containing various critical decryption keys were read (decrypted by another key derived from *one* manually-entered passphrase) and loaded into the memory to perform future cryptographic services.

Four or five of us on the engineering team knew the passphrase and could restart cryptoserv if it crashed or simply had to have an upgrade. What if someone performed a little old-fashioned rubber-hose cryptanalysis and literally beat the passphrase out of one of us? The attacker could theoretically get access to these all-important master keys. Then stealing the encrypted-at-rest database of all our users' secrets could prove useful – they could decrypt them in the comfort of their underground supervillain lair.

I *needed* to eliminate this threat.

Shamir Secret Sharing was the obvious choice – beautiful, simple, perfect (you can in fact prove that if done right, it offers perfect secrecy.) I decided on a 3-of-8 scheme and implemented it in pure POSIX C for portability over a few days, and tested it for several weeks on my Linux desktop with other engineers.

Step 1: generate the polynomial coefficients for 8 shard-holders.

Step 2: compute the key shards (x_0, y_0) through (x_7, y_7)

Step 3: get each shard-holder to enter a long, secure passphrase to encrypt the shard

Step 4: write out the 8 shard files, encrypted with their respective passphrases.

And to reconstruct:

Step 1: pick any 3 shard files.

Step 2: ask each of the respective owners to enter their passphrases.

Step 3: decrypt the shard files.

Step 4: reconstruct the polynomial, evaluate it for $x=0$ to get the key.

Step 5: launch cryptoserv with the key.

One design detail here is that each shard file also stored a message authentication code (a keyed hash) of its passphrase to make sure we could identify when someone mistyped their passphrase. These tests ran hundreds and hundreds of times, on both Linux and Solaris, to make sure I did not screw up some big/little-endianness issue, etc. It all worked perfectly.

A month or so later, the night of the key splitting party was upon us. We were finally going to close out the last vulnerability and be secure. Feeling as if I was about to turn my fellow shard-holders into *cymeks*, I gathered them around my desktop as PayPal's front page began sporting the "*We are down for maintenance and will be back soon*" message around midnight.

The night before, I solemnly generated the new master key and securely copied it to cryptoserv. Now, while "Push It" by Salt-n-Pepa blared from someone's desktop speakers, the automated deployment script copied shard files to their destination.

While each of us took turns carefully entering our elaborate passphrases at a specially selected keyboard, Paul shut down the main database and decrypted the payment credentials table, then ran the script to re-encrypt with the new key. Some minutes later, the database was running smoothly again, with the newly encrypted table, without incident.

All that was left was to restore the master key from its shards and launch the new, even *more* secure cryptographic service.

The three of us entered our passphrases... to be met with the error message I haven't seen in weeks: "*Sorry, one or more wrong passphrases. Can't reconstruct the key. Goodbye.*" Surely one of us screwed up typing, no big deal, we'll do it again. No dice. No dice – again and again, even after we tried numerous combinations of the three people necessary to decrypt.

Minutes passed, confusion grew, tension rose rapidly.

There was nothing to do, except to hit rewind – to grab the master key from the file still sitting on cryptoserv, split it again, generate new shards, choose passphrases, and get it done. Not a great feeling to have your first launch go wrong, but not a huge deal either. It will all be OK in a minute or two.

A cursory look at the master key file date told me that no, it wouldn't be OK at all. The file sitting on cryptoserv *wasn't* from last night, it was created just a few minutes ago. During the Salt-n-Pepa-themed push from stage, we overwrote the master key file with the stage version. Whatever key *that* was, it wasn't the one I generated the day before: only one copy existed, the one I copied to cryptoserv from my computer the night before. *Zero* copies existed now. Not only that, the push script appears to have also wiped out the backup of the *old* key, so the database backups we have encrypted with the old key are likely useless.

Sitrep: we have 8 shard files that we apparently cannot use to restore the master key and zero master key backups. The database is running but its secret data cannot be accessed.

I will leave it to your imagination to conjure up what was going through my head that night as I stared into the black screen willing the shards to *work*. After half a decade of trying to make something of myself (instead of just going to work for Microsoft or IBM after graduation) I had just destroyed my first successful startup in the most spectacular fashion.

Still, the idea of "what if we all just continuously screwed up our passphrases" swirled around my brain. It was an easy check to perform, thanks to the included MACs. I added a single `printf()` debug statement into the shard reconstruction code and instead of printing out a summary error of "one or more..." the code now showed if the passphrase entered matched the authentication code stored in the shard file.

I compiled the new code directly on cryptoserv in direct contravention of all reasonable security practices – what did I have to lose? Entering my own passphrase, I promptly got "bad passphrase" error I just added to the code. Well, that's just great – I knew *my* passphrase was correct, I had it written down on a post-it note I had planned to rip up hours ago.

Another person, same error. Finally, the last person, JK, entered his passphrase. No error. The key still did not reconstruct correctly, I got the "Goodbye", but something worked. I turned to the engineer and said, "what did you just type in that worked?"

After a second of embarrassed mumbling, he admitted to choosing "a\$\$word" as his passphrase. The gall! I asked everyone entrusted with the grave task of relaunching cryptoserv to pick *really* hard to guess passphrases, and this guy...?! Still, this was something – it worked. But *why*?!

I sprinted around the half-lit office grabbing the rest of the shard-holders demanding they tell me their passphrases. Everyone else had picked much lengthier passages of text and numbers. I manually tested each and none decrypted correctly. *Except* for the a\$\$word. What was it...

A lightning bolt hit me and I sprinted back to my own cubicle in the far corner, unlocked the screen and typed in "*man getpass*" on the command line, while logging into cryptoserv in another window and doing exactly the same thing there. I saw exactly what I needed to see.

Today, should you try to read up the programmer's manual (AKA the *man page*) on `getpass`, you will find it has been long declared obsolete and replaced with a more intelligent alternative in nearly all flavors of modern Unix.

But back then, if you wanted to collect some information from the keyboard without printing what is being typed in onto the screen and remain POSIX-compliant, `getpass` did the trick. Other than a few standard file manipulation system calls, `getpass` was the *only* operating system service call I used, to ensure clean portability between Linux and Solaris.

Except it wasn't completely clean.

Plain as day, there it was: the manual pages were identical, except Solaris had a "special feature": any passphrase entered that was longer than 8 characters long was automatically reduced to that length anyway. (Who needs long passwords, amiright?!)

I screamed like a wounded animal. We generated the key on my Linux desktop and entered our novel-length passphrases right here. Attempting to restore them on a Solaris machine where they were being clipped down to 8 characters long would never work. Except, of course, for `a$$word`. That one was fine.

The rest was an exercise in high-speed coding and some entirely off-protocol file moving. We reconstructed the master key on my machine (all of our passphrases worked fine), copied the file to the Solaris-running cryptoserv, re-split it there (with very short passphrases), reconstructed it successfully, and PayPal was up and running again like nothing ever happened.

By the time our unsuspecting colleagues rolled back into the office I was starting to doze on the floor of my cubicle and that was that. When someone asked me later that day why we took so long to bring the site back up, I'd simply respond with "eh, shoulda RTFM."

RTFM indeed.

P.S. A few hours later, John, our General Counsel, stopped by my cubicle to ask me something. The day before I apparently gave him a sealed envelope and asked him to store it in his safe for 24 hours without explaining myself. He wanted to know what to do with it now that 24 hours have passed.

Ha. I forgot all about it, but in a bout of "what if it doesn't work" paranoia, I printed out the base64-encoded master key when we had generated it the night before, stuffed it into an envelope, and gave it to John for safekeeping. We shredded it together without opening and laughed about what would have never actually been a company-ending event.

P.P.S. If you are thinking of all the ways this whole SSS design is horribly insecure (it had some real flaws for sure) and plan to poke around PayPal to see if it might still be there, don't. While it served us well for a few years, this was the very first thing eBay required us to turn off after the acquisition. Pretty sure it's back to a single passphrase now.

Notes:

1: a member of Chicagoland sci-fi fan community let me know that the original news of our move to the US was delivered to them via a posted letter, *snail mail*, not FidoNet email!

Posted 1 week ago

403 notes 5 Comments

Like 6 Share

 dancinglaughingforgetting liked this


 frxstguardian liked this

 wuggen liked this


 this-sure-is-a-blog reblogged this from blue-mystique

 bluekryptoniteobservation liked this

 cyreneduvent reblogged this from max-levchin


 chiops256 reblogged this from max-levchin

 chiops256 liked this

 revcleo reblogged this from max-levchin

 revcleo liked this

 babybat98 reblogged this from tathrin

 tathrin reblogged this from catchaspark

 clearestbluest liked this

 quietmarie reblogged this from max-levchin

 quietmarie liked this

 bre1000m reblogged this from max-levchin

 laghairt-dragon liked this


 aviaryavidity liked this

 avenir77 liked this

 acetrainerjess reblogged this from monikoishi

 acetrainerjess liked this

 monikoishi reblogged this from ariamaki


 ariamaki reblogged this from max-levchin

 ariamaki liked this

 cowboy-dreamin reblogged this from starfoozle


 namingcrisis liked this

 stellara-ferrium liked this

 fingors reblogged this from notarealwelder

 miraclequorra reblogged this from photomatt

 blue-mystique reblogged this from photomatt

 patchwork-passions liked this

 tathrin liked this

 wingweaver24 liked this

 pascalize liked this

 procyon-potor reblogged this from catchaspark

 thegreatcatwar reblogged this from max-levchin

 thegreatcatwar liked this

 forgettamouse liked this

 antiquery liked this

 morhath liked this

 uwgaragestudios liked this

 ethereal-call liked this

 contrapositive-post-posting reblogged this from andmaybegayer

 kreuz-unlimited reblogged this from nightpool

 kreuz-unlimited liked this

 creepylookingthang reblogged this from max-levchin

 ajhalili2006 liked this

 marbled-b0nes liked this

 rhewkath reblogged this from bluef00t

Show more notes
